J. Elsing H. Sterner A. Wagner

BASIC AUF DEM COMMODORE 64



BASIC-Einführung und Erläuterung spezifischer Eigenschaften

iwt

J. Elsing H. Sterner A. Wagner

BASIC AUF DEM COMMODORE 64



BASIC-Einführung und Erläuterung spezifischer Eigenschaften

iwt

CIP-Kurztitelaufnahme der Deutschen Bibliothek

Elsing, Jürgen:
BASIC auf dem Commodore 64: BASIC-Einf. u.
Erl. spezif. Eigenschaften / J. Elsing;
H. Sterner; A. Wagner. — 1. Aufl. — Vaterstetten
bei München: IWT, 1983.
ISBN 3-88322-029-9

NE: Sterner, Heinz; Wagner, Annette:

ISBN 3-88322-029-9
1. Auflage 1983

Umschlag: Kaselow + Partner
Holdenrieds Druck- und Verlags-GmbH, Füssen

Vorwort

Dieses Buch bietet eine systematische Einführung in die Programmiersprache BASIC für den Commodore 64. Außer vielen kleineren Programmen zur Illustrierung der einzelnen BASIC-Anweisungen findet der Benutzer eine umfangreiche Programmsammlung zu den verschiedensten Themenbereichen.

Die besonderen Fähigkeiten des Commodore 64 werden mit vielen Beispielprogrammen erläutert.

Für denjenigen, der die zum Teil recht umfangreichen Programme nicht alle selbst eingeben möchte, stehen die Programme auch auf Kassette oder Diskette zur Verfügung.

Wir hoffen, daß Ihnen dieses Buch möglichst viele Fragen, die sich bei der Benutzung Ihres Computers ergeben werden, beantworten kann.

Für alle Hinweise, Tips, kritische (aber natürlich auch zustimmende) Reaktionen, die einer Verbesserung dieses Buches dienen können, sind wir dankbar!

Die Autoren

Vorwort 5

Einleitung 11

1. Allgemeine Grundlagen der Programmierung 15

Von der Problembeschreibung zum Programm 15
Die grundlegenden Elemente zur Darstellung von Algorithmen 19
Darstellung von Flußdiagrammen 19

2. Grundlegende BASIC-Anweisungen 25

Ausgabe und arithmetische Operationen 25
Eingabe und Zuweisung 29
Verzweigungen 33
Speichern von Programmen 37

3. Weitere BASIC-Anweisungen 41

Daten im Programm 41
Wiederholte Vorgänge 43
Einfache Variablen 45
Unterprogramme 48
Weitere Verzweigungsmöglichkeiten 50
Tabellenverarbeitung mit Feldern 50
Eingebaute Funktionen 56
Formatierfunktionen 56
Numerische Funktionen 56
Textverarbeitung mit Stringfunktionen 59
Sonstige Funktionen 62
Logische Operationen 63

4. Speichern und Laden von Programmen und Daten auf Kassette und Diskette 69

Speichern von Programmen 69
Laden von Programmen 70
Speichern und Laden von Daten 71
Speichern von Daten: Kassettenbetrieb 71
Einlesen von Daten: Kassettenbetrieb 74
Speichern von Daten: Diskettenbetrieb 75
Einlesen von Daten: Diskettenbetrieb 76
Einfache Datenverwaltung auf der Diskette 77

5. Einführung in die Speicherorganisation des Commodore 64 85

Der RAM-Speicher 85
Der ROM-Speicher 85
Die Anweisungen POKE und PEEK 86
Bildschirm- und Farbspeicher 87
Der Bildschirmspeicher 87
Der Farbspeicher 88
Vordergrund- und Hintergrundfarbe 90
Der Mehrfarbenmodus 90

6. Hochauflösende Grafik 95

Sprites 95
Sprite-Übersicht 105
Hochauflösende Grafik 107

7. Musikprogrammierung 113

Grundlagen zur Musikprogrammierung 113
Tonhöhe, Lautstärke, Wellenform, Anschlag .. 113
Die Hüllkurve 118
Mehrstimmige Musikstücke 127
Übersicht über die Musikprogrammierung 138

8. Einige Tips und Tricks 143

Cursorsteuerung 143
Änderung des Zeichensatzes 148
Verschieben des Bildschirmspeichers 157
Verkleinern des BASIC-Speichers 159
Benutzung der Funktionstasten 160

9. Etwas Theorie 165

Bit, Byte und Nibble 165
Das Binär-System 166
High- und Low-Byte 167
Das Hexadezimal-System 169

10. Programmsammlung 175

Digitaluhr 176
Wochentagsbestimmung 182
Numerische Integration 184
Division mit N Stellen 192
Geburtstagswahrscheinlichkeit 194
Zahlenumrechnungen 196
Umrechnung von Binärzahlen in Dezimazahlen 196
Umrechnung von Dezimalzahlen in Binärzahlen 200
Umwandlung von Dezimalzahlen in Zahlen mit beliebiger Basis und umgekehrt 202
Logik-Formel 206
Lissajous-Figuren 208
Die Maus im Labyrinth 212
Monte Carlo - Berechnung von PI 218
LIFE 224
Zinseszins-Berechnungen 234
Sortier-Verfahren 238
Bubble-Sort 239
Sortieren durch Einfügen 243
SHELLSORT 247
QUICKSORT 254
17 und 4 262
NIM-Spiel 266
Schiffe versenken 270
Roulette 276
Wörterraten 280
Laufschrift 284
Tipp - Test 286
Lernkarten 288
Musik-Eingabe-Programm 290

Anhang 305

Speicherbelegung des Commodore 64 306
Der Bildschirmspeicher 310

Der Farbspeicher 312
Farbcodes 314
Der Grafikspeicher 316
Sprite-Datenblatt 318
Speicherübersicht: Sprites 320
Speicherübersicht: Musik 324
Frequenz-Tabellen 328
Die Commodore-Zeichen 332
Fehlermeldungen 336

Literaturverzeichnis 340
Stichwortverzeichnis 342

Einleitung

Das vorliegende Buch beginnt mit den allgemeinen Grundlagen der Programmierung und endet mit einer umfangreichen Sammlung von Programmen aus den unterschiedlichsten Bereichen. Dazwischen wird der Leser und Computerbenutzer Schritt für Schritt systematisch mit den Möglichkeiten der Programmiersprache BASIC sowie den speziellen Möglichkeiten des Commodore 64 vertraut gemacht.

Im ersten Kapitel werden grundlegende Begriffe wie Algorithmus, Flußdiagramm und Programm erläutert. Es wird dargestellt, daß man mit nur fünf verschiedenen Elementen im Grunde sämtliche Algorithmen formulieren kann. Darauf basiert dann auch die Einführung der grundlegenden BASIC-Anweisungen im zweiten Kapitel, in dem gezeigt wird, wie Ein- und Ausgabe, arithmetische Operationen sowie Zuweisungen und Verzweigungen in BASIC realisiert werden. Außerdem erfährt man, wie Programme auf der Kassette oder Diskette gespeichert werden können.

Im nächsten - 3. - Kapitel werden weitere BASIC-Anweisungen in einer solchen Reihenfolge eingeführt, daß eine neue Anweisung jeweils notwendig erscheint, um ein bestimmtes Problem zu lösen. In diesem Kapitel werden auch die eingebauten BASIC-Funktionen und die logischen Operationen besprochen. Im darauf folgenden vierten Kapitel wird noch einmal auf das Speichern und Laden von Programmen eingegangen. Zusätzlich wird an Beispielen aufgezeigt, wie Daten auf externe Speicher gebracht und auf einfache Weise kleine Dateien angelegt werden können.

Im fünften Kapitel gewinnt der Benutzer des Commodore 64 einen Einblick in die Speicherorganisation seines Computers. Unter anderem finden sich hier nähere Informationen zum Bildschirm- und Farbspeicher. Dazu werden die Anweisungen PEEK und POKE eingeführt, die in den noch folgenden Kapiteln häufig gebraucht werden.

Der Grafik-Programmierung ist ein eigenes Kapitel gewidmet, in dem die Erstellung von Sprite-Figuren und Darstellungen in hochauflösender Grafik (im Commodore-Handbuch überhaupt nicht besprochen) erklärt wird.

Im siebten Kapitel erhalten Sie einen Eindruck von den musikalischen bzw. akustischen Eigenschaften Ihres Computers. Da der Commodore 64 über einen speziellen Baustein für die Musikprogrammierung verfügt, lassen sich recht bemerkenswerte Ergebnisse erzielen - mit bis zu drei Stimmen übrigens.

In den folgenden 'Tips und Tricks' erhält man z.B ausführliche Hinweise über die Cursorsteuerung und die Benutzung der Funktionstasten. Außerdem wird gezeigt, wie man sich einen eigenen Zeichensatz erzeugen kann.

Im neunten Kapitel ergänzen einige theoretische Grundlagen das Wissen des inzwischen fortgeschrittenen Programmierers. Begriffe wie Bit und Byte, Binär- und Hexadezimal-Sytem sind ihm anschließend geläufig.

Die sich anschließende Programmsammlung enthält Anwendungs-Beispiele aus den verschiedensten Bereichen: neben Spielen, Simulationen, Zinseszins-Berechnungen, Zahlenumrechnungen und weiteren mathematischen Berechnungen findet sich hier ein Hilfsprogramm, das die Eingabe von Noten in Musikprogramme erheblich erleichtert. Außerdem werden mehrere Sortierverfahren (jeweils für Zahlen und für Strings) vorgestellt und miteinander verglichen.

Im Anhang werden nützliche tabellarische Übersichten über die einzelnen Speicherbereiche, Farbcodes, Commodorezeichen, Frequenztabellen usw. geboten. Außerdem finden Sie hier eine kommentierte Liste der Fehlermeldungen des Commodore 64.

Im Literaturverzeichnis werden Hinweise auf weitere Bücher zum Commodore 64 bzw. zur Programmierung in BASIC gegeben.

Ein Stichwortverzeichnis erleichert Ihnen das Auffinden gesuchter Begriffe.

1

Allgemeine Grundlagen

Von der Problembeschreibung zum Programm

Ein Computer unterscheidet sich von einem einfachen Rechner in erster Linie dadurch, daß er frei programmierbar ist. Ein Taschenrechner z.B. hat die Schrittfolge zur Berechnung einer bestimmten Rechenoperation 'fest eingebaut'.

Da den meisten Menschen der Umgang mit Taschenrechnern bereits sehr vertraut ist, denkt kaum jemand daran, daß sich bei einem einfachen Tastendruck recht komplexe Vorgänge abspielen. So erfordert z.B. die Berechnung einer Quadratwurzel eine Vielzahl von elementaren Anweisungen, die durch einen einzigen Tastendruck abgerufen werden.

Ein Programm ist nun nichts anderes als eine solche Folge von Anweisungen, die einem Rechner oder Computer mitteilen, was er tun soll.

Während aber ein einfacher Rechner nur eine ganz bestimmte Anzahl von Anweisungen 'versteht', (nämlich die auf den Tasten angegebenen Funktionen) ist ein Computer grundsätzlich nicht auf eine bestimmte Zahl von Anweisungen festgelegt, sondern kann durch eine entsprechende Programmierung prinzipiell alle möglichen Anweisungen zur Verarbeitung von Daten (also Zahlen und Texten) ausführen, die eindeutig formuliert sind.

Grundsätzlich ist ein Computer auch keineswegs nur eine 'Rechenmaschine', sondern vielmehr eine 'Datenverarbeitungsmaschine', denn ein Computer kann nicht nur mit Zahlen umgehen, sondern ganz allgemein mit irgendwelchen Zeichen oder Symbolen. Das können Texte sein, Adresslisten oder beliebige andere Ansammlungen von Daten. Die 'Datenverarbeitung' kann im Berechnen, Zählen, Sortieren, Vergleichen usw. bestehen.

Da man einem Computer aber nicht einfach in der Umgangssprache mitteilen kann, welches Problem er lösen soll (dies ist - kurz gesagt - darin begründet, daß die Umgangssprache in vielen Fällen nicht eindeutig ist),

muß man sich spezieller '<u>Programmiersprachen</u>' bedienen, die auf einem begrenzten Wortschatz mit einer genau festgelegten Syntax (Satzaufbau) basieren, die der Computer 'verstehen' kann. (Verstehen heißt in diesem Zusammenhang einfach, daß der Computer imstande ist, eine bestimmte Anweisung in einer Programmiersprache auf eine ganz bestimmte Weise auszuführen.)

Die Programmiersprache, die Sie im vorliegenden Buch kennenlernen sollen, heißt <u>BASIC</u>. Das ist eine Abkürzung für "Beginner's All Purpose Symbolic Instruction Code". BASIC wurde aus einer der ältesten Programmiersprachen - nämlich FORTRAN - entwickelt und sollte insbesondere zu Lernzwecken dienen. Die Sprache gewann aber rasch eine weitaus größere Verbreitung als vorgesehen und wurde fortwährend verändert und erweitert. Deshalb existieren eine ganze Reihe von BASIC-'Dialekten', die sich in unterschiedlichem Maße voneinander unterscheiden. D.h. daß die BASIC-Version, die auf Ihrem Computer läuft, nicht unbedingt ohne Änderungen auf einem anderen Computer-Typ laufen muß und umgekehrt.

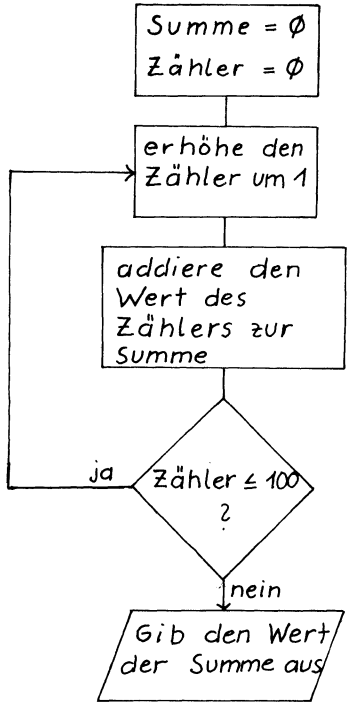
Da ein Computer also nur 'seine' Programmiersprache versteht, muß man sich vor dem Schreiben eines Programmes genau überlegen, wie das spezielle Problem in dieser Sprache formuliert werden muß. Unabhängig von einer Programmiersprache ist es aber generell zuvor notwendig, das Problem in Einzelschritte zu zerlegen.

Man kann also einem Computer nicht etwa einfach befehlen: "Zähle die Zahlen von 1 bis 100 zusammen!", sondern man müßte ihm - in der richtigen Reihenfolge - die dazu notwendigen einzelnen Schritte angeben.

Vorausgesetzt, daß unser Computer mit den folgenden Anweisungen etwas anfangen kann, könnten diese Anweisungen z.B. wie folgt aussehen:

1. Setze die Summe auf 0.
2. Setze einen Zähler auf 0.
3. Erhöhe den Zähler um 1.
4. Addiere den Wert des Zählers zur Summe.
5. Ist der Wert des Zählers $\leq$ 100, so fahre fort mit Schritt 3, sonst mit Schritt 6.
6. Gib den Wert der Summe aus.

Um eine Aufgabe derart zu zergliedern und trotzdem noch übersichtlich darzustellen, eignet sich am besten eine grafische Darstellung, die den Ablauf der einzelnen Schritte veranschaulicht. Eine solche Darstellung nennt man dementsprechend Ablauf- oder Flußdiagramm. In unserem Beispiel könnte das Flussdiagramm so aussehen:



Damit haben wir auch gleich einen wichtigen Begriff anhand eines Beispiels erklärt, nämlich den Begriff '<u>Algorithmus</u>':

Ein Flußdiagramm ist nämlich nichts anderes als eine grafische Darstellung eines Algorithmus, also einer Folge von Anweisungen, die (wenn sie entsprechend detailliert formuliert sind) es ermöglichen, ein bestimmtes Problem zu lösen.

Die Darstellung der einzelnen Schritte eines Algorithmus hängt natürlich ganz davon ab, für wen die Anweisungen bestimmt sind. Will man die Lösung einer Aufgabe einem Computer übertragen, so kann man dem Computer nicht direkt ein Flußdiagramm eingeben, sondern man muß vielmehr den Algorithmus in eine andere Form bringen, indem man ihn als Programm in einer Programmiersprache formuliert, die der Computer 'verstehen' kann. Ein Programm ist also - ebenso wie ein Flußdiagramm - eine bestimmte Darstellung eines Algorithmus.

Fassen wir noch einmal zusammen:

Ein Algorithmus ist eine Folge von Anweisungen, die in einer bestimmten Reihenfolge abgearbeitet werden müssen, um zu einem vorgegebenen Ziel zu gelangen.

Die Darstellung eines Algorithmus kann auf verschiedene Weise erfolgen, z.B. grafisch in Form eines Flußdiagramms.

Für den Computer muß ein Algorithmus jedoch mit den Mitteln einer Programmiersprache formuliert werden, die der Computer 'versteht', d.h. deren Anweisungen er ausführen kann.

Zur Erstellung eines Programms gehören also folgende Schritte:

1. Analyse des Problems (wie läßt sich das Problem zergliedern?).

2. Erstellung eines Programmablaufplans (Flußdiagramm).

3. 'Übersetzung' in die Programmiersprache.

Die grundlegenden Elemente zur Darstellung von Algorithmen

Die Hauptaufgaben einer Programmiersprache lassen sich unter den Begriffen

- Eingabe,
- Ausgabe,
- Zuordnung,
- Arithmetik (Rechenoperationen) und
- Verzweigung

zusammenfassen.

Da jede Programmiersprache über entsprechende Sprachmittel verfügt, könnte man grundsätzlich jedes Problem, für dessen Lösung sich ein Algorithmus angeben läßt, in eine beliebige Programmiersprache übertragen. Praktisch sieht es jedoch so aus, daß sich bestimmte Probleme in einer bestimmten Sprache besser darstellen lassen als in einer anderen. Dies erklärt auch die Vielzahl der existierenden Programmiersprachen.

Darstellung von Flußdiagrammen

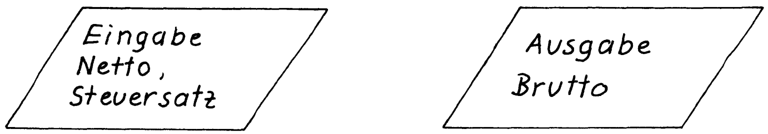
Im folgenden sollen die wichtigsten grafischen Elemente zur Darstellung von Flußdiagrammen vorgestellt werden.

Allgemein besteht ein Flußdiagramm aus einer Anzahl von grafischen Symbolen (z.B. Rechtecken), in die in freier Formulierung die einzelnen Anweisungen eines Algorithmus eingetragen werden. Der Programmablauf wird durch entsprechende Verbindungslinien gekennzeichnet, wobei der Hauptfluß des Programms von oben nach unten erfolgt. Abweichende Richtungsverläufe werden durch Pfeile markiert.

Ein- und Ausgabe

Als Symbol für eine Ein- oder Ausgabe verwendet man ein Parallelogramm, das die notwendigen Angaben über die Ein- bzw. Ausgabewerte enthält:

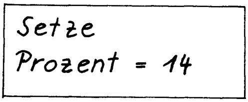
Beispiel:



Zuordnung und allgemeine Operationen (Arithmetik)

Das allgemeine Symbol für beliebige Operationen ist ein Rechteck:

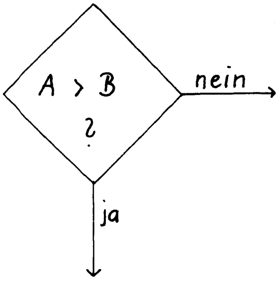
Beispiel:



Verzweigung

Verzweigungen ermöglichen es, je nach Ausgang einer Entscheidung, den Programmlauf an verschiedenen Stellen fortzusetzen:

Beispiel:



Programmanfang- und Ende

Programmanfang- und Ende werden durch das folgende Symbol gekennzeichnet. Eine solche Kennzeichnung mag auf den ersten Blick überflüssig erscheinen, ist jedoch zumindest bei umfangreicheren Flußdiagrammen notwendig.

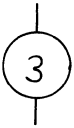
Beispiel:



Anschlussstellen

Nur kürzere Flußdiagramme lassen sich auf einer Seite unterbringen. Die 'Fortsetzungen' eines Programms werden durch Kreis-Symbole (am jeweiligen Programmende- bzw. Fortsetzungsstück) dargestellt, die eine beliebige Kennzeichnung enthalten (z.B. Zahlen).

Beispiel:



Bei der Entwicklung von Flußdiagrammen empfiehlt es sich, so vorzugehen, daß man das zu lösende Problem schrittweise verfeinert. D.h., man beginnt mit einer groben Skizze des Programmablaufs und entwickelt nach und nach die einzelnen Abschnitte im Detail. Dieses - unter dem Namen 'Strukturierte Programmierung' bekannte ' Verfahren ist in BASIC leider nur eingeschränkt möglich. (Moderne Sprachen wie Pascal eignen sich sehr viel besser für eine solche Programmentwicklung.)

2

BASIC

Generell findet jede Form der Datenverarbeitung in drei Schritten statt: die Daten, die verarbeitet werden sollen, müssen eingegeben werden; anschließend erfolgt die eigentliche Verarbeitung der Daten und schließlich die Ausgabe der Ergebnisse. Diese drei Schritte lassen sich also kurz durch die Begriffe: Eingabe, Verarbeitung, Ausgabe umreißen.

Ausgabe und arithmetische Operationen

Der wichtigste Verarbeitungsschritt ist in gewisser Hinsicht der zuletzt genannte, denn ohne eine Ausgabe von Ergebnissen ist die gesamte Datenverarbeitung vergebens.

Sehen wir uns also zunächst an, wie wir den Computer dazu bringen, daß er z.B. Texte oder Zahlen bzw. die Ergebnisse von Berechnungen auf dem Bildschirm anzeigt. Dazu dient der PRINT-Befehl, der - ganz allgemein gesagt - eine Ausgabe auf dem Bildschirm bewirkt.

PRINT

Vorweg noch ein Hinweis: viele BASIC-Anweisungen können nicht nur innerhalb eines Programmes stehen, sondern auch direkt benutzt werden. Was das genau bedeutet, probieren wir am besten gleich einmal mit dem PRINT-Befehl aus. Schreiben Sie z.B.:

```
PRINT "HALLO"
```

und drücken anschließend die RETURN-Taste, so erscheint das Wort HALLO (oder jeder andere von Ihnen gewählte kurze Text) auf dem Bildschirm. Der dem PRINT-Befehl folgende, von Anführungszeichen eingeschlossene Text wird also genauso wieder ausgegeben, wie er zwischen den Anführungszeichen steht. Dieser Text darf, bis auf Anführungszeichen, sämtliche möglichen Zeichen (auch Grafiksymbole der Tastatur!) enthalten. Die Anführungszeichen selbst dienen nur als Trennzeichen.

Die PRINT-Anweisung kann aber nicht nur Texte oder beliebige Zeichenketten unverändert ausgeben, sondern es lassen sich damit auch Rechenoperationen ausführen.

Geben Sie z.B. folgende PRINT-Anweisung ein:

```
PRINT 3+5
```

Nachdem Sie die Eingabe mit der RETURN-Taste abgeschlossen haben, sehen Sie sofort das Ergebnis in der nächsten Zeile.

Im Unterschied zu dieser direkten Eingabe werden die Anweisungen eines Programms erst dann ausgeführt, wenn das ganze Programm vollständig eingeben ist und gestartet wird. Woran erkennt der Computer, ob es sich um eine einzelne Anweisung handelt, die wir eingeben wollen, oder ob eine Anweisungsfolge, also ein Programm, eingegeben werden soll? Die Antwort läßt sich sehr schnell geben:

Eine Anweisung wird in BASIC dann als Bestandteil eines Programms angesehen, wenn sie mit einer Zeilennummer beginnt.

Geben Sie also einmal das folgende Programm ein:

```
10 PRINT "3+5="
20 PRINT 3+5
```

Anders als bei der direkten Eingabe dieser Anweisungen tut sich auf dem Bildschirm zunächst nichts. Da der Computer ja nicht 'weiß', ob noch weitere Anweisungen für dieses Programm folgen, kann er natürlich nicht einfach irgendwann mit der Programmausführung beginnen. Ein Programm muß also ausdrücklich gestartet werden. Dies geschieht mit dem Systembefehl RUN, der - wie jede Eingabe von der Tastatur (eine Ausnahme wird später besprochen) - mit RETURN abgeschlossen wird. (Im folgenden verzichten wir deshalb meistens auf den Hinweis, daß eine Tastatur-Eingabe mit der RETURN-Taste abgeschlossen werden muß.)

RUN

Nachdem Sie also das Programm mit RUN gestartet haben, erhalten Sie als Ausgabe zwei Zeilen: in der ersten steht die Zeichenfolge 3+5 (aus der PRINT-Anweisung in Programmzeile 10), darunter das Ergebnis der Addition von 3 und 5 (aus der PRINT-Anweisung in Programmzeile 20). Der 'Text' (bzw. die Zeichenfolge) des ersten PRINT-Befehls ist also genauso wiedergegeben worden, wie er zwischen den Anführungszeichen stand. Beim zweiten PRINT-Befehl - ohne Anführungszeichen -

wurde zunächst die Berechnung von 3+5 durchgeführt, und dann das Ergebnis (8) auf dem Bildschirm ausgegeben.

Vielleicht möchten Sie aber Text und Ergebnis nebeneinander statt untereinander auf dem Bildschirm sehen? Dann setzen Sie in Zeile 10 zusätzlich ein Semikolon an das Ende der Zeile. Dadurch schließt sich die Ausgabe des folgenden PRINT-Befehls unmittelbar an die vorhergehende Ausgabe an.

Wenn Sie statt eines Semikolons zum Abschluss der Zeile ein Komma verwenden, erfolgt die nächste PRINT-Ausgabe zwar auch in derselben Zeile, jedoch in einem bestimmten Abstand. Die Bildschirmzeile wird in dem Fall in vier Abschnitte eingeteilt, und die Ausgabe des folgenden PRINT-Befehls beginnt an der ersten Position des jeweils nächsten Viertels; also entweder in Spalte 1, 11, 21 oder 31.

Die PRINT-Anweisung hat also feste Tabulator-Positionen 'eingebaut', die es auf einfache Weise ermöglichen, Ausgaben in Tabellenform zu erstellen.

TAB

Allerdings ist man bei der Formatierung nicht auf diese Möglichkeiten beschränkt, sondern kann mit einer 'richtigen' Tabulatorfunktion die Ausgabe in einer beliebigen Spalte auf dem Bildschirm beginnen lassen. Dazu gibt man hinter dem Schlüsselwort TAB in einer PRINT-Anweisung in Klammern an, wieviele Spalten vor der Ausgabe übersprungen werden sollen. Durch die Anweisung

```
PRINT TAB(13);"TEST"
```

beginnt z.B. die Ausgabe des Wortes TEST in Spalte 14.

SPC

Während in der Funktion TAB absolute Bildschirm-Positionen angegeben werden, kann man mit der Funktion SPC eine bestimmte Anzahl von Leerzeichen zwischen beliebige Ausgaben setzen. Geben Sie z.B. ein:

```
PRINT "ERSTER";SPC(5);"VERSUCH"
```

so erscheint das Wort ERSTER zu Beginn der Zeile, darauf folgen 5 Leerzeichen sowie das Wort VERSUCH.

Wie Sie sehen, können in einer PRINT-Anweisung auch mehrere Ausdrücke stehen, die auszugeben sind. In diesen Fällen werden die einzelnen Ausdrücke durch Komma oder Semikolon voneinander getrennt.

Im folgenden Programm sind die Möglichkeiten der PRINT-Anweisung noch einmal zusammengestellt. Bevor Sie jedoch das Programm eingeben, müssen Sie das letzte Programm erst durch die Anweisung NEW löschen. Diese Anweisung sollten Sie aber nur dann benutzen, wenn das im Computer stehende Programm wirklich nicht mehr benötigt wird!

NEW

```
10 PRINT "DIE PRINT-ANWEISUNG"
20 PRINT
30 PRINT "2+3 =";5
40 PRINT "2+3 =";2+3
50 PRINT 1,2,3,4
60 PRINT "A";"B";"C";"D"
70 PRINT TAB(5);"TABULATOR";TAB(15);"TABULATOR"
80 PRINT TAB(5);"TABULATOR";SPC(15);"TABULATOR"
```

Nachdem Sie dieses Programm vollständig eingegeben und mit RUN gestartet haben, erhalten Sie folgende Ausgabe auf dem Bildschirm:

```
DIE PRINT-ANWEISUNG

2+3 = 5
2+3 = 5
 1         2         3         4
ABCD
     TABULATOR TABULATOR
     TABULATOR               TABULATOR

READY.
```

Zeile 10 bewirkt die unveränderte Ausgabe des in Anführungszeichen stehenden Textes.

Zeile 20 führt einen Zeilenvorschub aus, erzeugt also eine Leerzeile.

Zeile 30 gibt die in Anführungszeichen eingeschlossenen Zeichen sowie den Wert des darauf folgenden Ausdrucks aus, der in diesem Fall nur aus einer einzigen Zahl besteht.

Zeile 40 unterscheidet sich von der vorhergehenden Zeile dadurch, daß der - ohne Anführungszeichen angegebene! - arithmetische Ausdruck zunächst berechnet wird und dieser Wert dann ausgegeben wird.

Zeile 50 gibt vier Zahlen an den voreingestellten vier Tabulator-Positionen aus (Trennzeichen: Komma).

Zeile 60 bewirkt die Ausgabe von vier Buchstaben ohne Zwischenräume direkt nacheinander (Trennzeichen: Semikolon).

Zeile 70 schreibt das Wort 'Tabulator' zweimal in eine Bildschirmzeile: das erste Mal beginnt die Ausgabe in der 5., das zweite Mal in der 15. Spalte, wobei die Spaltennumerierung mit Null beginnt!

Zeile 80 unterscheidet sich von Zeile 70 dadurch, daß beim zweiten Mal die SPC-Funktion verwendet wird, sodaß das zweite Wort durch 15 Leerzeichen vom ersten Wort getrennt ausgegeben wird.

LIST

Übrigens: wenn Sie Ihr Programm wieder auf dem Bildschirm sehen wollen, geben Sie einfach den Befehl LIST ein, der bewirkt, daß alle im Computer stehenden Programmzeilen - nach den Zeilennummern sortiert - auf dem Bildschirm erscheinen. Möchten Sie nur eine Zeile sehen, so geben Sie nach LIST die gewünschte Zeilennummer an, z.B.: LIST 70. Entsprechend ist es auch möglich, einen bestimmten Zeilenbereich auszugeben, z.B. mit LIST 30-50.

Eingabe und Zuweisung

Bisher standen alle Daten, die der Computer für eine bestimmte Ausgabe benötigte, fest im Programm: in Form von Zeichenketten (Texten) oder in Form von Zahlen oder mathematischen Ausdrücken, die vor der Ausgabe berechnet wurden. Für eine neue Berechnung mit anderen Daten mußten die Daten im Programm geändert werden.

Diese Form der 'Eingabe' von Daten ist jedoch nur dann sinnvoll, wenn die betreffenden Daten gar nicht oder nur sehr selten geändert werden müssen. Eine wesentlich flexiblere Möglichkeit ist durch die Eingabe während des Programmablaufs - gewissermaßen im Dialog mit dem Computer - gegeben.

In BASIC dient die INPUT-Anweisung dazu, während des Programmablaufs Informationen an den Computer zu geben. Es versteht sich, daß die einzugebenden Werte nicht mehr wie bisher als 'konstante' Werte fest im Programm stehen können, denn das Programm selbst kann durch eine Eingabe während des Programmlaufs nicht verändert werden. In einem solchen Fall stehen im Programm statt der konstanten Werte 'Platzhalter' oder 'Variablen' für die einzugebenden Daten, die erst durch die Eingabe einen bestimmten Wert zugewiesen bekommen.

INPUT

Der Name einer solchen Variablen beginnt grundsätzlich mit einem Buchstaben. Als zweites Zeichen kann ein Buchstabe oder eine Ziffer folgen. Für die Lesbarkeit von Programmen sind zwar einerseits aussagefähige - und das heißt im allgemeinen: längere - Variablenbezeichnungen vorzuziehen, anderseits sind im BASIC des Commodore 64 nur die beiden ersten Zeichen eines Namens signifikant, d.h. der Computer ignoriert die übrigen Zeichen einfach, sodaß also z.B. MANN und MAUS als identische Variablen angesehen würden.

Angenommen, wir haben eine Reihe von Werten vorliegen, die Nettobeträge darstellen, und wollen nun die jeweilige Mehrwertsteuer und die Bruttobeträge als Summe aus Nettobetrag und Mehrwertsteuer berechnen, so könnte ein Programm dazu so aussehen:

```
10 PRINT "MEHRWERTSTEUER-BERECHNUNG"
20 PRINT
30 INPUT"NETTO =";NET
40 PRINT "MWST =";NET*0.14
50 PRINT "BRUTTO =";NET+NET*0.14
```

Neu ist an diesem Programm lediglich die INPUT-Anweisung in der Zeile 30. Dabei wurde gleich von der Möglichkeit Gebrauch gemacht, mit der INPUT-Anweisung zusätzlich einen erklärenden Text auszugeben, der (wie schon von der PRINT-Anweisung her gewohnt) ebenfalls in Anführungszeichen steht. Beachten Sie, daß zwischen diesem Text und der folgenden Variablen ein Semikolon stehen muß! Benutzt man die INPUT-Anweisung ohne Text, so entfällt das Semikolon.

Vielleicht hat es Sie schon gestört, daß in Zeile 50 noch einmal die gleiche Berechnung wie in Zeile 40 durchgeführt wird. (Natürlich hätte man den Brutto-Wert auch einfach durch Multiplikation des Netto-Wertes mit 1.14 erhalten können.) Wie könnte man eine solche doppelte Arbeit vermeiden? Nun, man müßte nach der ersten Berechnung den ermittelten Wert als Zwischenergebnis speichern, und ihn nachher wieder aus diesem Speicher abrufen. Woher nehmen wir den Speicher dafür? Dazu verwenden wir einfach wieder eine Variable, der wir den entsprechenden Wert zuweisen, z.B. so:

```
LET MWST = NET * 0.14
```

LET

Damit haben wir eine neue Anweisung - LET - eingeführt, die folgendes bewirkt: der rechts vom Gleichheitszeichen stehende Ausdruck wird berechnet (falls erforderlich), und das Ergebnis wird der links vom Gleichheitszeichen stehenden Variablen zugewiesen. In den meisten BASIC-Versionen kann man allerdings das Schlüsselwort LET weglassen: dies gilt auch für das Commodore-BASIC.

Wäre der Wert der Variablen NET z.B. gleich 100, so hätte die Variable MWST nach der Ausführung der LET-Anweisung den Wert 100*0.14, also 14.

Die folgende Programmversion macht von dieser Möglichkeit des Zwischenspeicherns Gebrauch:

```
10 PRINT "MEHRWERTSTEUER-BERECHNUNG"
20 PRINT
30 INPUT "NETTO:";NET
40 PRINT "MWST =";NET*0.14
50 LET BRUT = NET*1.14
60 PRINT "BRUTTO =";BRUT
```

Die bisher verwendeten Variablen (NET, BRUT, MWST) dienten dazu, jeweils einen Zahlenwert zu speichern; es ist aber auch möglich, beliebige Zeichenfolgen oder Texte unter einem Variablennamen zu speichern. Die entsprechenden Variablen unterscheiden sich von den numerischen Variablen dadurch, daß sie mit dem $-Zeichen enden, z.B. A$ oder NAME$. (Auch hier ist aber wieder zu berücksichtigen, daß der Computer nur die beiden ersten Zeichen des Variablennamens - sowie natürlich das $-Zeichen - beachtet!)

Außerdem ist es bei der Verwendung von Variablen wichtig zu wissen, daß die Bezeichnungen TI, TI$ und ST bereits für bestimmte Aufgaben des Computers reserviert sind und nicht als normale Variablennamen verwendet werden können (s. auch Kapitel 3 und 4).

In dem letzten Beispiel könnte man die Überschrift zunächst einer Stringvariablen (engl. string = Zeichenkette) zuweisen und anschließend den Wert dieser Variablen in einer PRINT-Anweisung ausgeben:

```
5 U$ = "Mehrwertsteuer-Berechnung:"
10 PRINT U$
```

Dieses Verfahren bringt im vorliegenden Programm-Beispiel zwar keinen Vorteil, doch können Sie sich sicher leicht vorstellen, daß die Sache schon anders aussieht, wenn dieser Text mehrfach im Laufe des Programms ausgegeben werden sollte. Ein weiterer Vorteil eines solchen Vorgehens ergibt sich bei Programmänderungen: in diesem Fall wäre nur eine einzige Zeile zu ändern, egal, ob die entsprechende Variable einmal oder hundertmal im Programm vorkommt.

Die Zuweisung zu einer Stringvariablen kann ebenso wie die zu einer numerischen Variablen auch über eine INPUT-Anweisung erfolgen. Das alte Programm-Beispiel kann z.B. so erweitert werden, das zuerst noch eine Artikel-Bezeichnung erfragt wird. Gleichzeitig wird in der folgenden Programm-Version von der Möglichkeit Gebrauch gemacht, mehrere BASIC-Anweisungen in eine Programmzeile zu schreiben, wobei die einzelnen Anweisungen durch einen Doppelpunkt getrennt werden:

```
5 U$="MEHRWERTSTEUER-BERECHNUNG"
10 PRINT U$
15 PRINT
20 INPUT "ARTIKEL ";A$
30 INPUT "NETTO =";NET: PRINT
40 PRINT A$;" MWST =";NET*0.14
50 LET BRUT=NET*1.14
60 PRINT A$;" BRUTTO =";BRUT
```

Wenn man mehrere Anweisungen in eine Programmzeile schreibt (vgl. Zeile 30), muß man die maximale Zeilenlänge einer Programmzeile beachten: im Commodore-BASIC darf eine BASIC-Zeile höchstens 80 Zeichen enthalten.

Da wir gerade bei Einschränkungen sind: eine Zeichenkette, also ein String, darf auch nicht beliebig lang sein, sondern kann nur bis zu 255 Zeichen enthalten. Sollte diese Länge überschritten werden, so gibt der Computer eine entsprechende Fehlermeldung aus: STRING TOO LONG (s. auch Anhang: Fehlermeldungen).

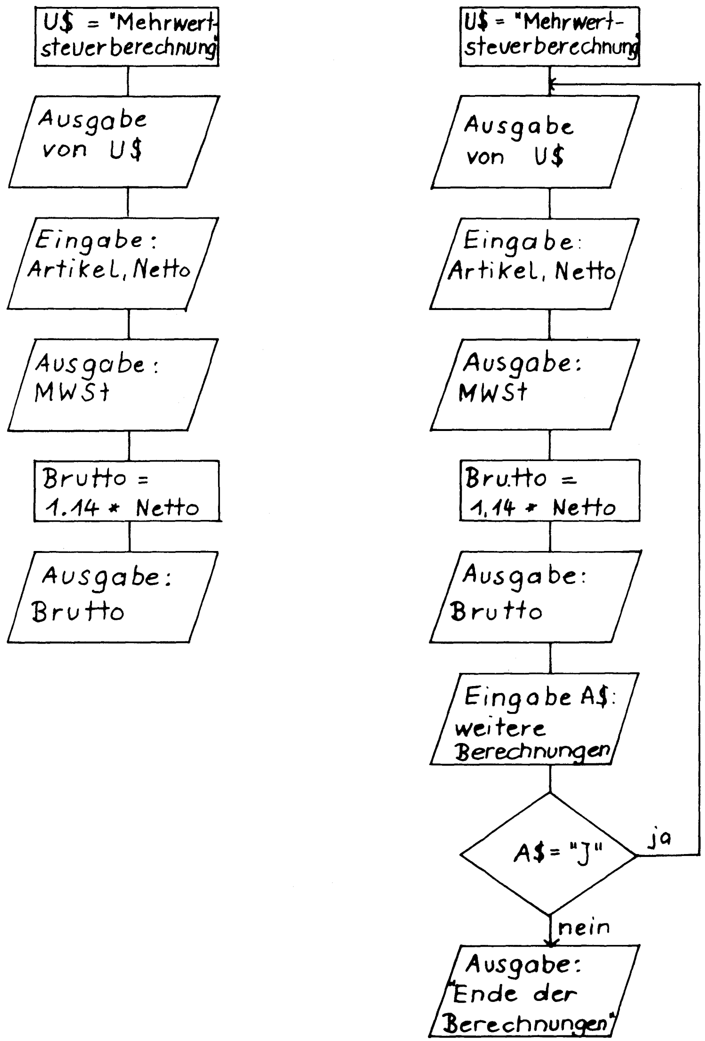
Während eine Eingabe über die INPUT-Anweisung immer mit der RETURN-Taste abgeschlossen werden muß, (wie alle übrigen Anweisungen auch) gibt es noch eine andere Eingabemöglichkeit, die es erlaubt, ein Programm mit nur einem Tastendruck zu steuern. Damit kommen wir - nach der Besprechung von Ein- und Ausgabe, Arithmetik und Zuweisung - auch schon zum letzten der fünf grundlegenden Elemente zur Darstellung von Algorithmen, zur Verzweigung.

V e r z w e i g u n g e n

Das bisher besprochene Programm-Beispiel läßt sich dadurch charakterisieren, daß die einzelnen Schritte einfach in einer eindeutig festgelegten Reihenfolge nacheinander abliefen, genau so, wie sie auch nacheinander im Programm stehen. Eine solche Programm-Struktur nennt man auch linear.

Die Grenzen eines linearen Programm-Ablaufs zeigen sich aber schon an unserem kleinen Beispiel-Programm: für jede neue Berechnung mußte das Programm immer wieder neu gestartet werden.

Dem Flußdiagramm der bisherigen Programm-Version ist im folgenden eine andere Fassung gegenübergestellt, die nach jeder Berechnung eine Abfrage enthält, ob das Programm fortgesetzt werden soll.



Im folgenden Programm sehen Sie, wie in BASIC eine Bedingung überprüft bzw. eine Entscheidung getroffen werden kann: Nach einer Berechnung wird das Programm nicht automatisch beendet, sondern über die INPUT-Anweisung wird gefragt, ob das Programm fortgesetzt werden soll (Zeile 60). Als Eingabe wird der Buchstabe J (für ja) bzw. N (für nein) erwartet und unter dem Namen der Variablen A$ gespeichert. In Zeile 70 wird dann in einer IF-Anweisung überprüft, ob das eingegebene Zeichen ein "J" ist. Nur dann, wenn diese Bedingung erfüllt ist, wird die auf das Schlüsselwort THEN folgende Anweisung ausgeführt. Ist die nach IF stehende Bedingung nicht erfüllt, wird das Programm sofort mit der nächsten Zeile fortgesetzt. Im Unterschied zu einem linearen Programmablauf kann ein Programm auf diese Art und Weise Verzweigungen enthalten.

IF ...

THEN ...

```
20 PRINT
25 INPUT "ARTIKEL ";A$
30 INPUT "NETTO =";NET:PRINT
40 PRINT A$;" MWST =";NET*0.14
45 LET BRUT=NET*1.14
50 PRINT A$;" BRUTTO =";BRUT
60 INPUT "WEITER (J/N)";A$
70 IF A$="J" THEN GOTO 10
80 PRINT "ENDE DER BERECHNUNGEN!"
```

Die auf das Schlüsselwort THEN (in Zeile 70) folgende Anweisung ist ebenfalls neu: GOTO 10 ist ein unbedingter Sprungbefehl, der bewirkt, daß das Programm mit der angegebenen Zeilennumnmer - also 10 in diesem Fall - fortgesetzt wird. Steht GOTO hinter der Anweisung THEN, reicht es allerdings aus, nur die Zeilennummer anzugeben.

GOTO

Wie bereits erwähnt, gibt es noch eine andere Möglichkeit, Daten während des Programmlaufes einzugeben, wobei die Eingabe nicht mit der RETURN-Taste abgeschlossen werden muß. Es handelt sich um die GET-Anweisung, die nur ein Zeichen von der Tastatur annimmt und danach sofort die Programmausführung fortsetzt. Wie bei der INPUT-Anweisung

GET

wird das eingegebene Zeichen unter dem Namen der nach GET stehenden (numerischen oder String-) Variablen gespeichert und kann danach zur Steuerung des Programmablaufes benutzt werden.

Abgesehen davon, daß mit der GET-Anweisung nur ein Zeichen eingegeben werden kann, unterscheidet sie sich auch dadurch von der INPUT-Anweisung, daß kein Text mit ausgegeben werden kann; dies muß vorher mit PRINT geschehen. Da in unserem Programm-Beispiel nur ein Zeichen einzugeben ist, kann die INPUT-Anweisung wie folgt durch GET ersetzt werden:

```
60 GET A$
65 IF A$="" THEN GOTO 60
```

Bei dieser Version genügt es dann, zur Fortsetzung des Programmes nur die J-Taste zu drücken; jede andere Eingabe beendet das Programm wieder. In den Zeilen 60 und 65 wird die Tastatur solange abgefragt, bis eine Taste gedrückt wird. In dem Fall wird der Variablen A$ das Zeichen dieser Taste zugewiesen und das Programm mit der folgenden Anweisung (hier also 70) fortgesetzt.

Die GET-Anweisung ist also (in Verbindung mit der IF-Anweisung) besonders gut dazu geeignet, Programme mit minimalem Tipp-Aufwand von der Tastatur her zu steuern.

Mit den BASIC-Anweisungen, die Sie bisher kennengelernt haben, können Sie nun schon grundsätzlich eine Vielzahl von kleineren Aufgaben programmieren: Sie wissen, wie Sie Daten in den Computer hineinbekommen und wie bestimmte Verarbeitungen (z.B. Zuordnungen oder Berechnungen) möglich sind. Sie kennen den Unterschied zwischen einem linearen und einem nichtlinearen Programmablauf und wissen, wie man ein Programm durch einen einzigen Tastendruck 'im Dialog' mit dem Computer steuern kann.

Es gibt jedoch noch eine ganze Reihe von weiteren BASIC-Anweisungen, die weitaus leistungsfähigere Programme ermöglichen. Diese Anweisungen sollen im folgenden Kapitel - illustriert durch kleine Beispielprogramme vorgestellt werden.

Speichern von Programmen

Falls Sie jetzt schon Programme schreiben, die Sie auch abspeichern und später wieder laden wollen, benötigen Sie dazu folgende Anweisungen (nähere Erläuterungen folgen in Kapitel 4).

Wenn Sie einen Kassettenrecorder benutzen, können Sie ein Programm durch:

```
SAVE "Name"
```

abspeichern und mit:

```
LOAD "Name"
```

wieder in den Arbeitsspeicher des Rechners laden. "Name" ist dabei der Name, den Sie für Ihr Programm gewählt haben. Er darf nicht mehr als 16 Zeichen enthalten und muß in Anführungszeichen stehen.

Die entsprechenden Anweisungen für ein Diskettenlaufwerk lauten:

```
SAVE "Name",8
```

bzw.:

```
LOAD "Name",8
```

```
  F
  O
P R I N T
O     E
K     X
E   S T E P
    H     E
    I     E X P
    F     K
    T A B
      S
      C H R $
          E
          S I N
          T
    G O T O
          R E A D
          E
```

3

Weitere BASIC-Anweisungen

Weitere BASIC-Anweisungen

Im zweiten Kapitel haben Sie bereits grundlegende Elemente der Programmiersprache BASIC kennengelernt, und Sie wissen inzwischen auch aus dem vorigen Kapitel, wie Sie Programme dauerhaft auf einer Kassette oder Diskette speichern können, um sie später wieder zu benutzen.

In diesem Kapitel sollen nun alle bisher noch nicht besprochenen BASIC-Anweisungen, die der Commodore 64 versteht, vorgestellt werden.

D a t e n   i m   P r o g r a m m

READ

DATA

Das nächste kleine Programm läßt sich als Telefonverzeichnis oder für ähnliche Anwendungen verwenden. Wie bekommen wir die Daten dafür in den Computer? Bei der Eingabe über INPUT wurden die Daten während des Programmlaufs eingegeben - und waren verschwunden, wenn das Programm gelöscht wurde. Die Anweisungen READ und DATA, die immer zusammen verwendet werden müssen, können unser Problem lösen. Betrachten Sie dazu das folgende Programm:

```
10 REM TELEFONVERZEICHNIS
20 INPUT"NAME: ";N$
30 READ X$,T
40 IF X$="*" THEN GOTO 80
50 IF X$<>N$ THEN GOTO 30
60 PRINT N$,T
70 GOTO 90
80 PRINT "NAME NICHT VORHANDEN!"
90 INPUT"WEITER (J/N)";A$
100 IF A$="J" THEN RESTORE: GOTO 20
110 END
120 REM.............DATEN
130 DATA MICHAEL,70214,HANS,55383,BEATE,42719
140 DATA ANNE,35126,KARL,71742
150 DATA *,0
```

Gleich in der ersten Zeile finden Sie eine neue Anweisung, REM (engl.: remark, Bemerkung), die zur Kennzeichnung einer Kommentarzeile dient: alles, was in der betreffenden Zeile hinter dieser Anweisung steht, wird vom Computer ignoriert. REM-Zeilen dienen also ausschließlich der Information des Benutzers. Machen Sie trotzdem ausgiebig Gebrauch von dieser Möglichkeit, Ihr Programm zu dokumentieren, da Sie bei längeren Programmen sonst schon nach kurzer Zeit selbst nicht mehr wissen, was das Programm eigentlich tun sollte.

REM

Der READ-Befehl in Zeile 30 bewirkt nun folgendes: Der erste Wert aus einer DATA-Zeile wird der hinter READ stehenden Variablen zugewiesen. Befinden sich - wie in unserem Fall - mehrere Variablen (durch Komma getrennt) hinter der READ-Anweisung, so wird der zweiten Variablen der zweite Wert aus den DATA-Zeilen zugewiesen usw. Jede weitere READ-Anweisung liest dann die folgenden Daten ein.

Nachdem das erste Datenpaar eingelesen wurde, wird zunächst überprüft, ob das Ende der Liste schon erreicht wurde, d.h. ob der zuletzt gelesene "Name" ein "*" war. Wenn ja, ist der gesuchte Name nicht vorhanden, und eine entsprechende Meldung wird in Zeile 80 ausgegeben. Ist das Ende der Liste noch nicht erreicht, so wird in Zeile 50 der mit READ aus der DATA-Zeile gelesene Name mit dem in N$ gespeicherten eingegebenen Namen verglichen. Stimmen die beiden Namen nicht überein, so wird das nächste Datenpaar gelesen usw. Fällt dieser Vergleich jedoch positiv aus, ist der gesuchte Name gefunden und wird zusammen mit der dazugehörigen Telefonnummer (T) ausgegeben.

Durch die Abfrage in Zeile 90 gesteuert, kann man beliebig oft nach einem Namen suchen. Durch RESTORE (Zeile 100) wird der DATA-"Zeiger" jedesmal wieder auf den Anfang der DATA-Zeilen gesetzt, damit der Rechner die Liste immer wieder von vorne durchlesen kann.

RESTORE

DATA-Zeilen können übrigens an beliebiger Stelle im Programm stehen, also auch vor einer READ-Anweisung und auch nach einer END-Anweisung, die zur Kennzeichnung des Programmendes dient. Diese Anweisung kann zwar in den meisten Fällen entfallen, nämlich dann, wenn das Programm mit der

END

letzten Zeile endet; man benötigt sie jedoch in solchen Fällen, in denen das Programm auf Grund von Verzweigungen nicht mit der letzten Zeile beendet ist.

Wenn Sie dieses Programm nun speichern, können Sie Ihr Telefon- oder Adressverzeichnis jederzeit wieder in den Computer einladen und benutzen. Wie Sie gesehen haben, spielt die Reihenfolge der Daten überhaupt keine Rolle, da gegebenenfalls immer die ganze Liste durchsucht wird.

W i e d e r h o l t e V o r g ä n g e

Ein Computer bietet sich in geradezu idealer Weise dazu an, sich wiederholende Berechnungen durchzuführen. Zur Illustration wollen wir ein Programm machen, das eine Tabelle der Quadrat- und Kubikzahlen für die Zahlen von 1 bis 10 erstellt:

```
10 REM TABELLE DER QUADRAT- UND KUBIKZAHLEN
20 PRINT " X","X↑2","X↑3": REM UEBERSCHRIFT
30 X=0
40 X=X+1
50 PRINT X,X*X,X↑3
60 IF X<10 THEN GOTO 40
70 END
```

Wie Sie sehen, dient das ↑-Zeichen zur Darstellung der Exponentenschreibweise. Das Programm würde um keine einzige Zeile länger, wenn die entsprechende Tabelle für 100, 1000 oder noch mehr Zahlen berechnet werden sollte, da dies nur von einem einzigen Wert abhängig ist: der Zahl in Zeile 60, die angibt, wie oft der Programmteil von Zeile 40 an durchlaufen werden soll. (Bei diesen Berechnungen dürften eigentlich nur ganzzahlige Ergebnisse auftreten; davon abweichende Werte sind Rundungsfehler.)

Für solche Fälle, wenn man also weiß, wie oft ein bestimmter Programmabschnitt wiederholt werden soll, gibt es noch eine besondere BASIC-Anweisung, die außerdem noch weitere Möglichkeiten bietet. Mit Hilfe dieser Anweisung wollen wir unser letztes Programm so ändern, daß unsere Tabelle nur die Quadrat- und Kubikwurzeln der geraden Zahlen von 2 bis 20 enthält:

```
10 REM TABELLE DER QUADRAT- UND KUBIKZAHLEN
20 PRINT " X","X*X","X↑3": REM UEBERSCHRIFT
30 FOR X=2 TO 20 STEP 2
40 PRINT X,X*X,X↑3
50 NEXT X
60 END
```

Die Programm-"Schleife" wird nun durch die Anweisungen:

FOR variable = anfangswert TO endwert STEP schrittweite

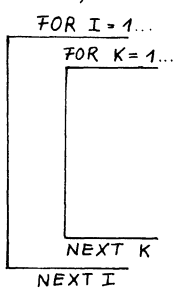
NEXT variable

FOR ... TO

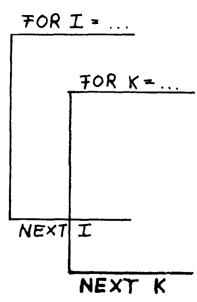
NEXT

realisiert, d.h. die in der Schleife stehenden Zuweisungen werden so oft wiederholt, bis die Laufvariable (hier: X) den hinter TO angegebenen Endwert erreicht hat. Dabei wird der Anfangswert bei jedem Durchgang um die hinter STEP angegebene Schrittweite erhöht. Die Angabe 'STEP Schrittweite' kann entfallen, wenn die Schrittweite gleich eins ist. Die Laufvariable dient aber nicht nur als Zähler, sondern kann gleichzeitig für Berechnungen innerhalb der Schleife verwendet werden, wie das Beispiel zeigt.

FOR-Anweisungen dürfen zwar ineinander verschachtelt sein, sich jedoch nicht überkreuzen:



Erlaubt!



Verboten!

Außerdem ist zu beachten, daß in eine Schleife nicht hineingesprungen und der Wert der Laufvariablen innerhalb der Schleife nicht verändert werden darf.

E i n f a c h e   V a r i a b l e n

Sie kennen schon zwei Typen von Variablen: die einen dienen zur Speicherung von Zahlen, die anderen zur Speicherung von Zeichenketten (oder Strings). Für beide gilt, daß ein Variablenname mit einem Buchstaben beginnen muß und das nächste Zeichen ein weiterer Buchstabe oder eine Ziffer sein darf. Man kann zwar längere Variablennamen benutzen, muß dann jedoch beachten, daß sich alle Namen in den beiden ersten Zeichen unterscheiden, da nur diese vom Computer ausgewertet werden.

Beispiele für zulässige Wertzuweisungen sind:

```
A = 1
X1 = 12.8
ALPHA = -123.456      (ALPHA = AL!)

A$ = "1"
G$ = "GUTEN TAG!"
```

Die durch das $-Zeichen gekennzeichnete <u>Stringvariable</u> A$ hat nichts mit der Variablen A zu tun! Und obwohl mit der Anweisung PRINT A,A$ zweimal die Ziffer '1' ausgegeben wird, handelt es sich nur im ersten Fall um die <u>Zahl</u> 1, im zweiten jedoch um den <u>String</u> "1", mit dem man natürlich z.B. keine Rechenoperationen vornehmen kann. Das können Sie leicht überprüfen, wenn Sie einmal versuchen, die Werte der Variablen A und A$ zu addieren. Das Resultat ist nicht etwa '2', sondern eine Fehlermeldung, weil Sie versucht haben, eine unzulässige Rechenoperation durchzuführen.

Die bisher bekannten numerischen Variablen können sowohl ganze Zahlen als auch Dezimalzahlen speichern. In manchen Fällen kann es aber ausgeschlossen werden, daß Dezimalzahlen überhaupt vorkommen (Telefonnummern z.B. können nur ganzzahlig sein). Dafür gibt es einen besonderen Variablentyp, der nur ganze Zahlen speichern kann. Die Kennzeichnung dieser Variablen erfolgt durch Anhängen des %-Zeichens an den Variablennamen. Diese Variablen werden als <u>Integer-Variablen</u> bezeichnet, wogegen die bisher bekannten <u>Real-Variablen</u> heißen.

Beispiele für die Zuweisung zu Integer-Variablen sind:

```
A% = 1
X1% = 12.8
ALPHA% = -1234
```

Wird einer Integer-Variablen ein nichtganzzahliger Wert zugewiesen, so wird dieser automatisch in eine ganze Zahl umgewandelt. Dies geschieht aber nicht durch Rundung der Zahl, sondern dadurch, daß die Nachkommastellen einfach abgeschnitten werden. Die Variable X1% hat also nach der obigen Zuweisung den Wert 12, wie Sie leicht feststellen können.

Da Integer-Variablen weniger Speicherplatz benötigen als Real-Variablen, kann man durch ihre Verwendung eventuell viel Speicherplatz sparen. Dies gilt vor allem dann, wenn es sich nicht um einfache Variablen handelt, die also nur einen Wert aufnehmen können, sondern um indizierte oder Feld-Variablen, die im übernächsten Abschnitt besprochen werden.

Der geringere Speicherplatzbedarf für Integer-Variablen hat natürlich auch seinen Preis. Dieser besteht darin, daß der Wertebereich für diesen Typ nur von -32767 bis 32767 reicht.

Real-Variablen hingegen können Zahlen im Bereich von $\pm$ 1.701411183E+38 und $\pm$ 2.93873588E-39 speichern.

Diese Zahlendarstellung, die zunächst etwas verwirrend aussehen mag, ist die wissenschaftliche oder exponentielle (daher der Buchstabe E) Notation. Dies ist nichts anderes als eine abkürzende Schreibweise, die insbesondere zur Darstellung sehr großer bzw. sehr kleiner Zahlen benutzt wird. Das Zeichen E darin steht für 'Exponent zur Basis 10', wobei mit Exponent die darauf folgende Zahl gemeint ist. Die Zahl vor dem Buchstaben E ist ein Faktor, mit dem das Ergebnis der Exponentiation multipliziert werden muß. Das hört sich vielleicht kompliziert an, tatsächlich aber ist das Umrechnen von der exponentiellen zur "normalen" Dezimaldarstellung (und umgekehrt) ziemlich einfach.

Betrachten Sie dazu einmal die folgende Gegenüberstellung einiger Zahlen in dezimaler und exponentieller Schreibweise:

Dezimaldarstellung	Exponentielle Notation	
1	1E0	$= 1 * 10\uparrow 0$
10	1E1	$= 1 * 10\uparrow 1$
100	1E2	$= 1 * 10\uparrow 2$
123	1.23E2	$= 1.23 * 10\uparrow 2$
123000000000	1.23E11	$= 1.23 * 10\uparrow 11$
.1	1E-1	$= 1 * 10\uparrow -1$
.01	1E-2	$= 1 * 10\uparrow -2$
.001	1E-3	$= 1 * 10\uparrow -3$
.00123	1.23E-3	$= 1.23 * 10\uparrow -3$
.00000000000123	1.23E-11	$= 1.23 * 10\uparrow -11$

Die Umrechnung einer Zahl aus der Exponential- in die Dezimalschreibweise ist einfach rein mechanisch so möglich:

Ist der Exponent (die Zahl hinter E) positiv, so erhält man die dezimale Notation, indem man das Komma bzw. den Dezimalpunkt der vor E stehenden Zahl um die im Exponenten angegebene Zahl nach rechts verschiebt. Gegebenenfalls ist eine entsprechende Anzahl von Nullen anzufügen.

Ist der Exponent negativ, so handelt es sich um eine Zahl, die kleiner ist als 1. Zur Umrechnung in die Dezimalschreibweise wird das Komma um die im Exponenten angegebene Zahl nach links verschoben, wobei die notwendige Anzahl von Nullen einzufügen ist.

Kehrt man diese Berechnungen sinngemäß um, so erhält man aus der dezimalen die exponentielle Notation: Von links beginnend stellen die von Null verschiedenen Zahlen den Faktor für den Exponentialausdruck dar. Dieser Faktor (auch Mantisse genannt) erhält einen Dezimalpunkt nach der ersten Stelle. Hinter dem Zeichen E wird dann als Exponent die Zahl notiert, die angibt, um wieviele Stellen das Komma nach rechts bzw. links verschoben wurde.

U n t e r p r o g r a m m e

Häufig kommt es vor, daß man an verschiedenen Stellen eines Programms die gleiche Anweisungsfolge benötigt. Es wäre nicht sehr wirtschaftlich, diese gleichartigen Programmteile mehrmals im Computer zu speichern (abgesehen von dem Schreibaufwand, der beim Programmieren entsteht). Deshalb schreibt man den entsprechenden Programmteil nur einmal in Form eines Unterprogramms, das dann vom Hauptprogramm aus "aufgerufen" wird. Wenn das Unterprogramm seine Aufgabe erledigt hat, wird das Programm an der Stelle fortgesetzt, von der aus der Aufruf des Unterprogramms erfolgte.

Durch die Verwendung von Unterprogrammen wird ein Programm jedoch nicht nur kürzer, sondern auch übersichtlicher. Das folgende Programm macht von der Unterprogrammtechnik Gebrauch:

```
10 REM ADDITION ZWEIER GANZER ZAHLEN
20 PRINT "1. ZAHL:"
30 GOSUB 200
40 N1=N
50 PRINT "2. ZAHL:"
60 GOSUB 200
70 N2=N
80 PRINT "SUMME:"
90 GOSUB 200
100 IF N=N1+N2 THEN PRINT "RICHTIG!": GOTO 20
110 PRINT "FALSCH!": GOTO 80
200 REM....UNTERPROGRAMM ZUR EINGABE
210 INPUT N
220 IF N<>INT(N) THEN PRINT"NUR GANZE ZAHLEN!": GOTO 210
230 RETURN

READY.
```

Das Programm erwartet die Eingabe zweier ganzer Zahlen sowie deren Summe. Hat man die Summe richtig angegeben, wird man mit der Antwort "RICHTIG!" belohnt, ansonsten erfolgt die Meldung "FALSCH!", und das Programm fragt solange nach der Summe, bis die richtige Zahl angegeben wird. Da keine Bedingung für ein Beenden des Programms eingebaut wurde, kann es nur mit der RUN/STOP- und RESTORE-Taste abgebrochen werden.

Der Aufruf eines Unterprogramms erfolgt mit der Anweisung GOSUB, gefolgt von der Zeilennummer, an der

GOSUB

RETURN

das Unterprogramm beginnt. Der Rücksprung zum aufrufenden Programmteil wird einfach durch die Anweisung RETURN bewirkt. Der Computer 'merkt' sich jeweils die Stelle, von der aus eine Verzweigung ins Unterprogramm erfolgte.

Die Eingabe sämtlicher Zahlen erfolgt in dem Unterprogramm, das bei Zeile 200 beginnt. Es ist zu beachten, das alle im Unterprogramm benutzten Variablen im gesamten Programm gelten. Will man in BASIC ein bereits geschriebenes Unterprogramm in ein bestehendes Programm einbauen, muß man also selbst gegebenenfalls für eine entsprechende Umbenennung der Variablennamen sorgen. Außerdem kann es erforderlich sein, die Zeilennumerierung zu ändern.

Wie stellt der Computer übrigens fest, ob es sich bei der eingegebenen Zahl um eine ganze Zahl handelt? Die entsprechende Überprüfung erfolgt in Zeile 220:

```
220 IF N<>INT(N) THEN PRINT"NUR GANZE ZAHLEN!":GOTO 210
```

Die ins BASIC 'eingebaute' Standardfunktion INT macht aus der in Klammern als Argument angegebenen Zahl (es kann auch ein Ausdruck sein, der erst noch berechnet werden muß, wie z.B.: 2 + 3.4) eine ganze Zahl,

INT

indem einfach der Nachkommateil abgeschnitten wird. Dabei erfolgt keine Rundung des Ergebnisses. Aus 5.01 wird also 5, ebenso aber auch aus 5.89. Wie man diese Funktion aber auch zum Runden von Rechenergebnissen verwenden kann, erfahren Sie bei der Besprechung der "Eingebauten Funktionen" im weiteren Verlauf dieses Kapitels.

Weitere Verzweigungsmöglichkeiten

Möchte man in Abhängigkeit vom Wert einer Variablen jeweils zu verschiedenen Programmteilen verzweigen, kann man natürlich eine entsprechende Anzahl von IF-Anweisungen verwenden, die je nach Ausgang des Vergleichs die gewünschte Verzweigung bewirken, z.B.:

ON GOTO

ON GOSUB

```
IF N = 1 THEN GOTO 100
IF N = 2 THEN GOTO 150
......

IF N = K THEN GOTO ...
```

Diese Anweisungen lassen sich vereinfachen zu:

```
ON N GOTO 100,150, ...
```

Dabei wird in Abhängigkeit des Wertes N=1,2,3, ... das Programm mit der 1., 2., 3. usw. hinter GOTO stehenden Zeilennummer fortgesetzt.

Eine entsprechende Möglichkeit gibt es auch für die Verzweigung zu verschiedenen Unterprogrammen. Sie lautet:

```
ON N GOSUB 100,150, ...
```

Hat N beispielsweise den Wert 2, so wird das Unterprogramm aufgerufen, das an 2. Stelle in der hinter GOSUB stehenden Liste angegeben ist, hier also 150.

Tabellenverarbeitung mit Feldern

Bisher haben wir nur mit einfachen Variablen zu tun gehabt, die als Platzhalter für einen einzigen Zahlenwert oder eine Zeichenkette (String) dienten. Es ist jedoch auch möglich, unter nur einem Namen mehrere Variablen anzusprechen, wobei die Unterscheidung dann durch die Angabe eines Index erfolgt.

Eine indizierte oder Feldvariable - oder kurz: ein Feld - ist im Grunde nichts anderes als eine Tabelle, die ja auch nur eine Bezeichnung hat. Die Bezugnahme auf einen bestimmten Wert aus einer Tabelle erfolgt dadurch, daß man die Zeile angibt, in der der betreffende Wert steht. Hat die Tabelle mehrere Spalten, so ist zusätzlich die Angabe der Spalte erforderlich.

Ein Beispiel soll dies veranschaulichen. Die monatlichen Umsätze eines Betriebes sollen für den Zeitraum eines Jahres in einer Tabelle dargestellt werden:

MONAT	UMSATZ
1	7000
2	7400
3	6800
4	6500
5	7600
6	7900
7	8200
8	8500
9	7800
10	7100
11	6400
12	6200

Nennen wir die Tabelle kurz UM, so können wir z.B. den 5. Wert mit UM(5) bezeichnen usw. Durch eine derartige Angabe ist ein Wert aus der Tabelle eindeutig bestimmt.

Wie speichert man eine solche Tabelle im Computer? Man kann natürlich für jeden Wert eine eigene Variable verwenden und etwa folgende Zuweisungen vornehmen:

```
10 REM TABELLE MIT EINFACHEN VARIABLEN
20 READ A,B,C,D,E,F,G,H,I,J,K,L
100 REM...............DATEN
110 DATA 7000,7400,6800,6500,7600,7900
120 DATA 8200,8500,7800,7100,6400,6200
```

Angenommen, wir wollen nun die Summe dieser Werte ermitteln. Also fügen wir folgende Zeilen ein:

```
30 SUM = A+B+C+D+E+F+G+H+I+J+K+L
40 PRINT "SUMME = ";SUM
```

Diese Aufgabe ist zwar mit einiger Schreibarbeit verbunden, aber es geht. Noch! - Denn stellen Sie sich vor, daß die Tabelle nicht 12, sondern 100, 1000 oder noch mehr Zahlen enthält: offensichtlich sind Sie dann sehr bald am Ende Ihrer Geduld und der Computer an den Grenzen seiner Speicherkapazität angelangt!

Wie Sie sich schon denken können, verwendet man in solchen Fällen keine einfachen, sondern Feldvariablen. In unserem Beispiel käme dabei folgendes heraus:

```
10 REM TABELLE MIT FELDVARIABLEN
20 DIM UM(12)
30 FOR I=1 TO 12
40 READ UM(I)
50 SUM = SUM+UM(I)
60 NEXT I
70 PRINT"SUMME =";SUM
100 REM....................DATEN
110 DATA 7000,7400,6800,6500,7600,7900
120 DATA 8200,8500,7800,7100,6400,6200

READY.
```

Was gibt es neues in diesem Programm? Vor der Verwendung einer Feldvariablen muß man dem Computer zu Beginn des Programms mitteilen, wie groß das betreffende Feld sein soll, damit eine entsprechende Zahl von Speicherplätzen dafür vorgesehen werden kann: das Feld muß "dimensioniert" werden. Die Anweisung dazu heißt DIM, gefolgt von dem Namen des Feldes, hinter dem in Klammern die Anzahl der zu reservierenden Speicherplätze steht.

DIM

Tatsächlich werden mit der Anweisung DIM UM(12) allerdings 13 Speicherplätze reserviert, da die Indexzählung mit Null beginnt, sodaß wir also auch der Variablen UM(0) einen Wert zuweisen könnten.

Notwendig ist eine Dimensionierung zwar erst dann, wenn der größte Index mehr als 10 beträgt, da der Computer bei Feldvariablen automatisch eine Dimensionierung bis 10 vornimmt; es ist jedoch empfehlenswert, in jedem Fall die Dimensionierung ausdrücklich anzugeben: zum einen spart man dadurch einigen Speicherplatz, zum anderen erhöht es die Übersicht über das Programm, wenn alle indizierten Variablen vorweg vereinbart werden.

Wie eine Tabelle nicht nur aus einer Spalte bestehen muß, können auch Feldvariablen mehr als eine Dimension haben. Die einzelnen Indizes werden dabei durch Komma getrennt angegeben. Z.B. bezeichnet A(2,5) die 2. Zeile, 5. Spalte einer 2-dimensionalen Tabelle. Die Dimensionierung erfolgt entsprechend durch die Angabe der beiden Indexgrenzen, z.B. DIM A(12,10). Beachten Sie, daß dadurch insgesamt (12+1) * (10+1) = 143 Speicherplätze reserviert werden! Mit ziemlich harmlos aussehenden DIM-Anweisungen kann man sehr schnell an die Grenzen der Speicherkapazität des Computers gelangen; so würde der Rechner etwa bei einer Dimensionierungs-Anweisung wie DIM A(500,100,10) einfach 'aussteigen', da er nicht annähernd soviele Speicherplätze besitzt (501*101*11 = 556611).

Das letzte Programmbeispiel soll jetzt so erweitert werden, daß wir mit den monatlichen Umsätzen von drei Betrieben Berechnungen anstellen wollen. Dazu legen wir die folgende Tabelle zu Grunde:

MONAT	BETRIEB 1	BETRIEB 2	BETRIEB 3
1	7000	6500	8100
2	7400	6900	8200
3	6800	7300	8500
4	6500	7900	9100
5	7600	7300	9500
6	7900	7200	9900
7	8200	8000	8800
8	8500	7300	8200
9	7800	6700	7900
10	7100	6600	7400
11	6400	6300	7200
12	6200	5800	6900

Die Aufgabe soll nun darin bestehen, die drei Jahressummen (je Betrieb) sowie die 12 monatlichen Summen aus den drei Betrieben zu ermitteln.

Außer dem zweidimensionalen Feld UM, das die Daten aus der Tabelle aufnehmen soll, werden zwei weitere Felder dimensioniert, die für die Speicherung der Jahressummen (JS) bzw. Monatssummen (MS) benutzt werden.

Das Programm liest zunächst sämtliche Daten aus den DATA-Zeilen ein (Zeile 30 - 80). Es wäre natürlich möglich, schon beim Einlesen Berechnungen vorzunehmen, der Übersicht halber geschieht dies jedoch jeweils getrennt in den beiden folgenden Programmteilen.

In den Zeilen 100 - 160 werden die drei Jahressummen berechnet. Deshalb läuft die äußere FOR-Schleife von 1 bis 3, entsprechend den drei Spalten der Tabelle. Für jede Spalte muß 12mal der entsprechende Zeilenwert aufaddiert werden, um die Jahressumme zu erhalten: diese Aufgabe wird durch die innere Schleife (Index I) erledigt. Nach der Berechnung jeder der drei Summen wird der jeweilige Betrag ausgegeben (Zeile 150).

Auf entsprechende Weise werden die 12 Monatssummen berechnet (Zeile 200 - 260). Der Unterschied zur vorigen Berechnung liegt nur darin, daß die Indizes vertauscht sind, da die Daten ja nun zeilenweise aufaddiert werden sollen.

```
10 REM ZWEIDIMENSIONALE TABELLENBERECHNUNG
20 DIM UM(12,3),JS(3),MS(12)
30 REM..................EINLESEN DER DATEN
40 FOR I=1 TO 12
50 FOR J=1 TO 3
60 READ UM(I,J)
70 NEXT J
80 NEXT I
100 REM.......................JAHRESSUMMEN
105 PRINT"JAHRESSUMMEN"
110 FOR J=1 TO 3
120 FOR I=1 TO 12
130 JS(J)=JS(J)+UM(I,J)
140 NEXT I
150 PRINT JS(J)
160 NEXT J
200 REM.......................MONATSSUMMEN
205 PRINT"MONATSSUMMEN"
210 FOR I=1 TO 12
220 FOR J=1 TO 3
230 MS(I)=MS(I)+UM(I,J)
240 NEXT J
250 PRINT MS(I)
260 NEXT I
300 REM..............................DATEN
310 DATA 7000,6500,8100
320 DATA 7400,6900,8200
330 DATA 6800,7300,8500
340 DATA 6500,7900,9100
350 DATA 7600,7300,9500
360 DATA 7900,7200,9900
370 DATA 8200,8000,8800
380 DATA 8500,7300,8200
390 DATA 7800,6700,7900
400 DATA 7100,6600,7400
410 DATA 6400,6300,7200
420 DATA 6200,5800,6900
```

Eingebaute Funktionen

Bis hierher haben Sie schon sämtliche BASIC-Anweisungen des Commodore 64 kennengelernt. Zusätzlich verfügt der Computer noch über "eingebaute" Funktionen zum Formatieren, für numerische Berechnungen, zur Textverarbeitung usw.

Einige Standardfunktionen - wie INT, TAB und SPC - haben wir schon verwendet. In diesem Abschnitt weroen sämtliche Standardfunktionen, nach Funktionsgruppen geordnet, vorgestellt.

Formatierfunktionen

POS(X)
Diese Funktion ergibt die Position in der Bildschirmspalte (0 - 39), an der die nächste Ausgabe mit PRINT ausgeführt würde.

SPC(X)
Diese Funktion erzeugt X Leerzeichen (space).

TAB(X)
Tabulatorfunktion. Diese Funktion wird in PRINT-Befehlen benutzt. Das darauf folgende Zeichen wird in Spalte X ausgegeben.

Numerische Funktionen

ABS(X) Absolutwert
Ergibt den absoluten Wert einer Zahl. Das Ergebnis ist immer positiv oder 0.

ATN(X) Arcustangens
Ergibt den Winkel (im Bogenmaß), dessen Tangens X ist.

COS(X) Cosinus
Ergibt den Cosinus des Winkels X, der im Bogenmaß angegeben werden muß.

EXP(X) Potenzfunktion
Entspricht e↑x, wobei e = 2.71828183.

FN F(X)   Selbstdefinierte Funktion

Liefert das Ergebnis einer vom Benutzer mit DEF FN definierten Funktion mit einer Variablen.

INT(X)   Integer-Funktion

Ergibt den ganzzahligen Anteil einer Zahl X. Alle Stellen nach dem Dezimalpunkt werden abgeschnitten. Das Ergebnis wird nicht gerundet und ist immer kleiner oder gleich X. Das bedeutet, daß negative Zahlen dem Betrag nach größer werden, z.B. INT(-2.4) = -3.

Die INT-Funktion kann jedoch auch zum Runden von Zahlen verwendet werden. Die Variable S in dem folgenden Ausdruck gibt dabei die Zahl der Stellen hinter dem Dezimalpunkt an:

```
X = INT(X*10↑S+.5)/INT(10↑S+.5)
```

Eine schnellere Berechnung ermöglicht die folgende Formel:

```
X = INT(X*P+.5)/P
```

Mit P=10 wird auf eine Stelle, mit P=100 auf zwei, mit P=1000 auf 3 Stellen gerundet, usw.

LOG(X)   Natürlicher Logarithmus

Ergibt den natürlichen Logarithmus von X zur Basis e. Zur Umwandlung in den Zehnerlogarithmus muß das Ergebnis durch LOG(10) dividiert werden.

RND(X)   RANDOM-Funktion

Diese Funktion ergibt eine (Pseudo-)Zufallszahl zwischen 0 und 1. Die Folge der Zufallszahlen läßt sich durch das Argument X steuern.

Um bei jedem Programmlauf einen neuen Anfangswert zu bekommen, sollte die erste Zufallszahl mit RND(-TI) erzeugt werden. Anschließend sollte X = 0 oder positiv sein.

Durch einen negativen Wert von X wird eine neue Startzahl bestimmt, die immer dieselbe Zahlenfolge ergibt, falls diese mit RND(1) erzeugt werden. RND(0) ergibt jedesmal neue Folgen.

Zur Erzeugung von Zufallszahlen im Intervall zwischen N1 und N2 kann man die folgende Formel verwenden:

R = INT(RND(1)*N2)+N1

SGN(X) Signum-Funktion

Vorzeichen-Funktion. Der Wert dieser Funktion ist 1, wenn X positiv, 0, wenn X=0 und -1, wenn X negativ ist.

SIN(X) Sinus

Ergibt den Sinus des Winkels X, der im Bogenmaß anzugeben ist.

SQR(X) Quadratwurzel

Ergibt die Quadratwurzel von X, wobei X größer oder gleich 0 sein muß.

TAN(X) Tangens

Ergibt den Tangens des Winkels X, der im Bogenmaß anzugeben ist.

Textverarbeitung mit String-Funktionen

Strings (oder Zeichenketten) sind uns schon in PRINT- oder INPUT-Anweisungen begegnet. Sie dienten dazu, eine Ein- oder Ausgabe zu erläutern.

Ein String kann - ähnlich wie eine Zahl - auch einer Variablen zugewiesen werden. Stringvariablen unterscheiden sich von Variablen, in denen Zahlen gespeichert werden können dadurch, daß an den Variablennamen ein $-Zeichen angehängt ist.

Die Möglichkeiten der String-Verarbeitung in BASIC reichen aber sehr viel weiter. Es gibt eine ganze Reihe von speziellen Funktionen, die eigentlich alle möglichen Operationen mit Zeichenketten durchführen können, die man sich denken kann. Um einige Versuche mit diesen Funktionen anzustellen, gehen wir einmal von dem folgenden String aus:

```
A$ = "GUTEN TAG!"
```

Aus wieviel Zeichen besteht dieser String? Halt, Sie brauchen nicht selbst zu zählen! Das macht die Funktion LEN:

```
PRINT LEN(A$)
10
```

Die LEN-Funktion kann genauso die Länge eines direkt angegebenen Strings berechnen, der natürlich in Anführungszeichen stehen muß:

```
PRINT LEN("TAG!")
4
```

Sollten Sie den ASCII-Code des ersten Zeichens eines Strings benötigen, so steht dafür die Funktion ASC zur Verfügung:

```
PRINT ASC("TAG!")
84
```

Das Gegenstück zu dieser Funktion liefert das Zeichen, das zu einem gegebenen ASCII-Wert gehört:

```
PRINT CHR$(84)
T
```

Jetzt wollen wir einmal unseren oben definierten String A$ "zerpflücken" und aus dem "GUTEN TAG!" einen "GUTEN ABEND!" machen. Dazu nehmen wir die ersten sechs Zeichen von A$ und hängen daran den String "ABEND". Ach ja, das Ausrufungszeichen haben wir vergessen. Das entnehmen wir wieder aus dem String A$, also:

```
PRINT LEFT$(A$,6);"ABEND";RIGHT$(A$,1)
GUTEN ABEND!
```

Die Funktionen LEFT$ und RIGHT$ liefern also den linken bzw. rechten Teil des Strings, der als Argument der Funktion verwendet wird. Die als zweites Argument angegebene Zahl bestimmt dabei die Länge der Teilstrings.

Die folgenden Programmzeilen servieren unseren Gruß "scheibchenweise":

```
10 A$="GUTEN TAG!"
20 FOR N=1 TO LEN(A$)
30 PRINT LEFT$(A$,N)
40 NEXT N
```

```
G
GU
GUT
GUTE
GUTEN
GUTEN
GUTEN T
GUTEN TA
GUTEN TAG
GUTEN TAG!
```

Von links und rechts können wir also schon Teile eines Strings abschneiden. Die Funktion MID$ ermöglicht es, einen beliebigen Teilstring aus einer Zeichenkette "herauszuschneiden". Um die Zeichenfolge "TAG" aus unserem String A$ zu erhalten, müssen von der 7. Position an 3 Zeichen entnommen werden:

```
PRINT MID$(A$,7,3)
TAG
```

Strings lassen sich auch miteinander verknüpfen. Als Operationszeichen dafür dient das "+"-Zeichen. Das folgende Programm veranschaulicht dies:

```
10 A$="GUTEN TAG!"
20 INPUT"WIE HEISSEN SIE";N$
30 B$=A$+N$
40 PRINT B$
```

Manchmal ist es notwendig, eine Zahl in einen String umzuwandeln, z.B. für Formatierungszwecke. Dies geschieht mit der Funktion STR$:

```
PI$ = STR$(3.1415)
PRINT PI$, LEFT$(PI$,4)
3.1415     3.14
```

Das Gegenstück dazu ist die Umwandlung eines Strings in eine Zahl. Das geht natürlich nur dann, wenn der String aus Ziffern besteht. Die Funktion VAL durchsucht den gegebenen String vom ersten linken Zeichen an n ach rechts, bis das erste Zeichen gefunden wird, das keine Zahl darstellen kann. Die so ermittelte Zeichenfolge wird in eine Zahl umgewandelt:

```
PRINT VAL("007BOND")
7
```

Sonstige Funktionen

FRE(X)   Freier Speicherplatz

Die Anweisung PRINT FRE(X) ergibt die Anzahl freier Bytes im Speicher. Dabei spielt der Wert von X keine Rolle.

SYS X   Aufruf eines Maschinenprogramms

Diese Funktion bewirkt den Aufruf eines Maschinenprogramms, das an der Adresse X beginnt. Der Wert von X kann eine Zahl zwischen 0 und 65535 sein.

Als Beispiel kann der Aufruf SYS 64738 dienen. Mit diesem Befehl wird der Rechner in den Grundzustand versetzt, in dem er sich auch nach dem Einschalten befindet. Aber Vorsicht! Der Befehl sollte nur benutzt werden, wenn der Computer wirklich ausgeschaltet werden sollte, da das Programm und die Variablenwerte verloren gehen.

USR(X)   Aufruf eines Maschinenprogramms

Diese Funktion ruft ein Maschinenprogramm auf, dessen Startadresse in den Speicherplätzen 785 und 786 stehen muß. Der Parameter X wird an das Maschinenprogramm übergeben, das selbst einen Wert als Ergebnis an das BASIC-Programm zurückgibt.

WAIT X,Y,Z

Diese Funktion dient dazu, ein Programm solange anzuhalten, bis eine bestimmte Speicheradresse einen bestimmten Wert angenommen hat. Nach WAIT folgt eine Speicheradresse (X) und bis zu zwei Variablen.

Der Inhalt der angegebenen Adresse wird zuerst mit dem exklusiven OR mit dem dritten Wert Z verknüpft (sofern dieser angegeben ist). Anschließend findet eine AND-Verknüpfung mit dem zweiten Wert Y statt. Ist das Ergebnis Null, wird der Inhalt der Adresse weiterhin überprüft. Ist das Ergebnis nicht Null, wird das Programm mit der nächsten Anweisung fortgesetzt.

L o g i s c h e   O p e r a t i o n e n

Ist Ihnen schon aufgefallen, daß das Gleichheitszeichen für zwei ganz verschiedene Operationen benutzt wird? In einer LET-Anweisung bewirkt es die Zuweisung eines Wertes zu einer Variablen, in einer IF-Anweisung jedoch dient es dazu, einen Vergleich zwischen zwei Ausdrücken darzustellen.

Im zweiten Fall wird also keine Rechen- sondern eine logische Operation vollzogen. Es gibt noch eine Anzahl weiterer logischer Vergleichs-Operatoren, die wir zum Teil auch schon benutzt haben:

=	gleich,
<>	ungleich,
<	kleiner als,
>	größer als,
<=	kleiner oder gleich,
>=	größer oder gleich.

Außerdem ist es aber auch möglich, einzelne Werte oder ganze Ausdrücke mit Hilfe von logischen Operatoren so zu verknüpfen, daß das Ergebnis ein logischer Wert ist, der 'Wahrheitswert' der gesamten Verknüpfung. Es handelt sich dabei um die logischen Operatoren NOT, AND und OR (also: NICHT, UND und ODER).

Bevor Sie Ausdrücke mit diesen logischen Operatoren verknüpfen, müssen Sie wissen, daß es außer Real-, Integer- und Stringvariablen noch einen weiteren Variablentyp gibt, der sich allerdings äußerlich nicht von den Real-Variablen unterscheidet. Diese 'logischen' Variablen nehmen als Ergebnis einer logischen Operation nur zwei Werte an: 0 oder -1, wobei 0 für den logischen Wert 'falsch' und -1 für 'wahr' steht. (Einschränkend muß gesagt werden, daß die logischen Variablen in BASIC keinen ganz selbständigen Variablentyp darstellen, da diese Variablen im selben Programm auch als normale Real-Variablen benutzt werden können.)

Betrachten wir dazu ein Beispiel:

```
PRINT 2 = 2
-1
```

d.h. die 'Aussage', daß 2 = 2 ist, hat den Wahrheitswert -1, ist also wahr. Dagegen ergibt

```
PRINT 2 = 3
0
```

d.h. diese Aussage ist falsch.

Da es nur die beiden Wahrheitswerte 'wahr' und 'falsch' (bzw. 0 und -1) gibt, erhält man bei der Verneinung oder Negation einer Aussage den jeweils anderen Wert. In BASIC wird die Negation mit NOT durchgeführt:

```
PRINT NOT(2 = 3)
-1
```

Was passiert nun, wenn zwei Ausdrücke mit AND oder OR verknüpft werden? Dazu wieder ein Beispiel:

```
PRINT 2=2 AND 2=3
0
```

Hier wurde also eine wahre mit einer falschen Aussage durch AND verknüpft. Das Ergebnis lautet, daß diese Verknüpfung falsch ist, d.h. es ist nicht wahr, daß beide Aussagen richtig sind. Die AND-Verknüpfung liefert nur dann den Wert -1 (also 'wahr'), wenn <u>beide</u> Teilaussagen <u>wahr</u> sind.

Verknüpfen Sie nun die beiden Ausdrücke mit OR:

```
PRINT 2=2 OR 2=3
-1
```

Obwohl ein Ausdruck falsch ist, hat das Ergebnis den Wert 'wahr'. Die OR-Verknüpfung ergibt nur dann den Wert 0 (also 'falsch'), wenn <u>beide</u> Teilaussagen <u>falsch</u> sind.

Die Ergebnisse der Verknüpfung zweier Ausdrücke A und B mit den logischen Operatoren NOT, AND und OR für die möglichen Kombinationen der beiden Ausdrücke sind in der folgenden Tabelle dargestellt:

A	NOT A
-1	0
0	-1

A	B	A AND B
-1	-1	-1
-1	0	0
0	-1	0
0	0	0

A	B	A OR B
-1	-1	-1
-1	0	-1
0	-1	-1
0	0	0

Sollen insgesamt drei Ausdrücke bzw. Variablen miteinander verknüpft werden, so ergeben sich 2↑3 = 8 mögliche Kombinationen; bei 4 Variablen sind es 2↑4 = 16; und allgemein sind es bei N: 2↑N Kombinationen.

4

Speichern & Laden

Speichern und Laden von Programmen und Daten auf Kassette und Diskette

S p e i c h e r n   v o n   P r o g r a m m e n

Mit der Anweisung SAVE kann ein Programm, das sich im Arbeitsspeicher des Computers befindet, auf einem externen Speichermedium gespeichert werden. Je nachdem,

SAVE

ob das Programm auf einer Magnetband-Kassette oder auf einer Diskette aufgezeichnet werden soll, unterscheiden sich die zu dem Befehl SAVE notwendigen Angaben voneinander: während zum Speichern eines Programmes auf dem Kasssetten-Recorder lediglich der - frei gewählte - Name des Programmes (der aus bis zu 16 Zeichen bestehen darf) in Anführungszeichen angegeben werden muß, ist zum Speichern auf einer Diskette zusätzlich die Angabe der Gerätenummer 8 erforderlich. (Das Band hat übrigens die Gerätenummer 1, die jedoch nicht angegeben werden muß, da sie vom Rechner als Standardeinstellung angenommen wird. Beim Speichern auf Band darf sogar der Programm-Name fehlen. Dies ist jedoch nicht unbedingt empfehlenswert.)

Die verschiedenen Möglichkeiten der Verwendung der SAVE-Anweisung sind in der folgenden Übersicht zusammengestellt:

SAVE	speichert ein Programm ohne Namen auf Band,
SAVE"NAME"	speichert das Programm unter dem Namen NAME auf Band,
SAVE A$	speichert das Programm mit dem Namen auf Band, den die Variable A$ enthält,
SAVE"NAME",8	speichert das Programm unter dem Namen NAME auf Diskette.

Laden von Programmen

Die Anweisung zum Laden eines Programmes vom Band oder von der Diskette in den Computer heißt LOAD und wird ähnlich benutzt wie die Anweisung SAVE. Sie dient dazu, Programme und Daten von der Kassette oder Diskette in den Arbeitsspeicher des Computers zu übertragen.

LOAD

Die verschiedenen Möglichkeiten der Verwendung der LOAD-Anweisung sind im folgenden zusammengestellt:

LOAD	liest das nächste Programm von der Kassette,
LOAD"NAME"	sucht auf der Kassette das Programm mit dem Namen NAME und lädt es in den Arbeitsspeicher, sofern es gefunden wurde,
LOAD "A$"	lädt das Programm mit dem Namen, den die Variable A$ enthält,
LOAD"NAME",8	lädt das Programm mit dem Namen NAME von der Diskette,
LOAD"A$",1,1	lädt ein Maschinenspracheprogramm mit dem in A$ enthaltenen Namen vom Band.
LOAD"A$",8,1	lädt ein Maschinenspracheprogramm mit dem in A$ enthaltenen Namen von der Diskette.

In den beiden letzten Fällen sollte ein NEW-Befehl nach dem Einladen gegeben werden. Andernfalls würde der Versuch, danach ein BASIC-Programm zu starten, zum Abbruch mit der Fehlermeldung OUT OF MEMORY führen!

Mit der Anweisung VERIFY kann überprüft werden, ob ein Programm richtig abgespeichert wurde. Dazu wird das angegebene Programm mit dem im Arbeitsspeicher stehenden Programm verglichen. Ebenso wie bei SAVE und LOAD muß auch hier der Programmname und eventuell die Gerätenummer angegeben werden. Die folgenden Fälle sind also möglich:

VERIFY

VERIFY — sucht das nächste Programm auf der Kassette und vergleicht es mit dem Programm im Speicher,

VERIFY"NAME" — sucht das Programm "NAME" auf der Kassette und vergleicht es mit dem Programm im Speicher,

VERIFY"NAME",8 — prüft das Programm "NAME" auf der Diskette.

Es empfiehlt sich, die VERIFY-Anweisung nach dem Speichern eines jeden Programmes zu benutzen, um sicher zu gehen, daß das Programm ohne Fehler auf der Kassette bzw. Diskette aufgezeichnet worden ist.

Speichern und Laden von Daten

Speichern von Daten: Kassettenbetrieb

Eine der häufigsten Anwendungen für einen Computer ist sicherlich die Verwaltung von Daten. Zur Speicherung stehen - wie bei der Programmspeicherung - wieder die Kassette oder die Diskette zur Verfügung. Dadurch ist es möglich, wesentlich mehr an Daten abzuspeichern, als im Speicher des Computers stehen können.

Eine bestimmte Menge von zusammengehörenden Daten, die durch einen Namen eindeutig identifizierbar sind, nennt man Datei oder File (engl.). Dateien oder Files werden mit der Anweisung OPEN erzeugt (bzw. 'geöffnet', falls sie schon existieren). Zusätzlich muß der Computer noch wissen, daß die Daten auf einen externen Speicher geschrieben werden sollen. Dazu dient ein spezieller PRINT-Befehl. An einigen Beispielen werden wir diese Möglichkeiten erläutern.

OPEN

Zur Eröffnung einer Datei auf der Kassette kann z.B. die folgende Anweisung dienen:

OPEN X,1,1,"NAME"

Dabei ist X die logische Datei- oder Filenummer und kann einen Wert von 0 bis 255 annehmen.

1 ist die Gerätenummer; in diesem Fall die des Kassettenrecorders.

Die zweite 1 gibt an, daß Daten abgespeichert werden sollen.

Der Name darf wieder aus bis zu 16 Zeichen bestehen.

PRINT#

Nachdem mit der OPEN-Anweisung die Datei NAME eingerichtet worden ist, können mit einer speziellen Form der PRINT-Anweisung, nämlich mit PRINT# Daten auf der Kassette gespeichert werden:

PRINT#X

wobei X wieder die logische Filenummer ist.

Zur Verdeutlichung haben wir ein kleines Programm geschrieben, das eine beliebige Anzahl von Zahlen abspeichern kann. (Die inversen Grafikzeichen dienen zur Cursorsteuerung. Sie beeinflussen nur die Bildschirmausgabe und können hier weggelassen werden. Näheres dazu s. Kapitel 9.)

```
10 REM.........................ABSPEICHERN VON DATEN AUF CASSETTE
15 PRINT"██ █ ABSPEICHERN VON DATEN AUF CASSETTE "
20 PRINT"██ NAME DES DATENFILES";:INPUT DF$
25 PRINT"█ WIEVIELE DATEN SOLLEN GESPEICHERT"
30 INPUT" WERDEN";N
40 DIM A(N)
45 REM.........................................EINGEBEN DER DATEN
50 FOR I=1 TO N
60 PRINT"██ WERT";I;:INPUT A(I)
70 NEXT I
80 PRINT"██ █CASSETTE EINGELEGT? (J/N)█"
90 GET C$: IF C$="" OR C$<>"J" THEN GOTO 90
```

```
95 REM...................................ABSPEICHERN DER DATEN
100 OPEN1,1,1,DF$
105 PRINT#1,N
110 FOR I=1 TO N
120 PRINT#1,A(I)
130 NEXT I
140 CLOSE 1
150 END

READY.
```

Nun die Programmerklärung:

In Zeile 20 wird nach dem Namen des Datenfiles gefragt, und in Zeile 25 wird gefragt, wieviele Daten (Zahlen) eingegeben werden sollen.

In Zeile 40 wird die indizierte Variable A entsprechend der Anzahl der einzugebenden Werte dimensioniert.

In den Zeilen 50 - 70 werden die Daten eingegeben.

Die Zeilen 80 und 90 sollen Sie daran erinnern, ein Band einzulegen.

In Zeile 100 wird das Datenfile geöffnet, und in den Zeilen 105 bis 130 werden die Daten auf das Band übertragen.

Zeile 140 enthält das Gegenstück zur Anweisung OPEN, nämlich

CLOSE

womit die Datei wieder 'geschlossen' wird.

Einlesen von Daten: Kassettenbetrieb

Das Einlesen von Daten bewirkt eine modifizierte INPUT-Anweisung. Sie lautet:

INPUT# X

wobei X wieder die logische Filenummer ist.

Dazu muß die OPEN-Anweisung folgendermaßen lauten:

OPEN X,1,0,"NAME"

Im Unterschied zur OPEN-Anweisung zum Speichern muß hier als dritte Zahl eine 0 stehen. Damit wird dem Computer mitgeteilt, daß er Daten einlesen soll.

Auch die INPUT#-Anweisung läßt sich wieder am besten durch ein kleines Programmbeispiel erklären:

```
10 REM................................DATEN LESEN VON CASSETTE
20 PRINT"▒▒ ▒       DATEN LESEN VON CASSETTE       "
30 PRINT"▒▒ WIE HEISST DAS DATENFILE";:INPUT DF$
80 PRINT"▒▒ ▒ CASSETTE EINGELEGT? (J/N)"
90 GET C$
100 IF C$="" OR C$<>"J" THEN GOTO 90
110 REM......................................EINLESEN DER WERTE
120 OPEN 1,1,0,DF$
130 INPUT#1,N
140 DIM A(N)
150 FOR I=1 TO N
160 INPUT#1,A(I)
170 NEXT I
180 CLOSE 1
190 REM................................AUSGABE AUF DEM BILDSCHIRM
200 FOR I=1 TO N
210 PRINTI;" ▮. WERT =";A(I)
220 NEXT I
230 END

READY.
```

Nun wieder die Erläuterung:

In Zeile 30 wird nach dem Namen der einzulesenden Datei gefragt.

Die Zeilen 80 bis 100 dienen, wie gehabt, zur Erinnerung, das richtige Band einzulegen.

In Zeile 120 wird das Datenfile zum Lesen geöffnet.

In Zeile 130 wird die Anzahl der abgespeicherten Daten eingelesen, und in den Zeilen 150 bis 170 werden die Daten eingelesen.

In Zeile 180 wird das Datenfile geschlossen, und in den Zeilen 200 - 220 werden die Daten auf dem Bildschirm ausgegeben.

Auch die GET-Anweisung existiert in einer modifizierten Version als GET#. Dadurch können aus einer Datei alle Zeichen einzeln nacheinander gelesen werden. Damit können auch Steuer- und Kontrollzeichen, die z.B. als Trennzeichen für die abgespeicherten Werte notwendig sind, erfaßt werden.

GET#

Speichern von Daten: Diskettenbetrieb

Der wesentliche Unterschied beim Speichern von Daten auf die Diskette gegenüber dem Kassettenbetrieb besteht darin, daß die OPEN-Anweisung einige zusätzliche Angaben enthält. Betrachten Sie dazu das folgende Programm:

```
10 REM........................ABSPEICHERN VON DATEN AUF DISKETTE
20 PRINT"    SPEICHERN AUF DISKETTE        "
30 PRINT" WIE SOLL DAS DATENFILE HEISSEN";:INPUT DF$
35 PRINT" WIEVIELE DATEN SOLLEN GESPEICHERT"
40 PRINT" WERDEN";: INPUT N
```

```
45 REM.............................................DATENEINGABE
50 DIM A(N)
55 FOR I=1 TO N
60 PRINT"▒ WERT";I;:INPUT A(I)
65 NEXT I
70 PRINT"▒▒ ▒ DISKETTE EINGELEGT (J/N)? "
75 GET C$
80 IF C$="" OR C$<>"J" THEN GOTO 75
85 REM.........................................DATEN SPEICHERN
90 OPEN 13,8,2,DF$+",S,W"
95 PRINT#13,N
100 FOR I=1 TO N
105 PRINT#13,A(I)
110 NEXT I
115 CLOSE13
120 END

READY.
```

Zeile 90 enthält die geänderte OPEN-Anweisung:

90 OPEN 13,8,2,DF$+",S,W"

Dabei bedeuten:

13	logische Filenummer,
8	Geräteadresse,
2	hier kann eine Zahl zwischen 2 und 14 stehen,
DF$	Filename,
S	sequentiell abspeichern,
W	WRITE-(Schreib-)Anweisung.

Einlesen von Daten: Diskettenbetrieb

Das Einlesen von Daten von der Diskette in den Computer erfolgt, bis auf eine geänderte OPEN-Anweisung, ebenso wie beim Kassettenbetrieb. Die OPEN-Anweisung für die Diskette unterscheidet sich von der zum Speichern verwendeten nur durch die Angabe R für READ (Lesen):

OPEN 13,8,2,DF$+",S,R"

Einfache Datenverwaltung auf der Diskette

Im nächsten Beispiel stellen wir Ihnen eine einfache Anwendung des bisher besprochenen vor. Wir greifen dazu das Telefonnummernprogramm aus Kapitel 3 wieder auf, und ändern es so um, daß dieses Verzeichnis jetzt auf einer Diskette abgespeichert werden kann.

Die Daten stehen dann also nicht mehr im Programm selbst, sondern als sequentielles Datenfile auf der Diskette. Bei solchen Files ist es möglich, weitere Daten an die schon vorhandenen einfach anzuhängen.

Diese Art der Datenspeicherung ist allerdings für eine solche Anwendung nur bei wenigen Daten sinnvoll, da Sie bei sequentiellen Dateien nicht direkt z.B. auf den 50. Datensatz zugreifen können (zum Lesen oder Ändern), sondern Sie immer alle Werte von Anfang an nacheinander lesen lassen müssen. (Im Unterschied dazu kann man bei einer Datei mit 'direktem Zugriff' - einer sog. 'Random Acces'-Datei - direkt auf einen beliebigen Datensatz zugreifen, ohne die ganze Datei durchlesen zu müssen).

In unserem Beispielprogramm ist vorgesehen, Name, Vorname, Vorwahl und Rufnummer einer Person abzuspeichern. Sie brauchen nun aber nicht Ihr ganzes privates Telefonverzeichnis in einem Arbeitsgang einzugeben, sondern können es später nach Belieben vervollständigen. Das wird ermöglicht durch den Befehl:

```
OPEN 1,8,2,"TELEFON,S,A"
```

Bis auf das "A" ist dies der schon bekannte OPEN-Befehl zum Anlegen bzw. Öffnen einer Datei. "A" besagt nun, daß hier keine neue Datei eröffnet werden soll, sondern Daten an ein bestehendes File (hier mit dem Namen "TELEFON") angehängt werden sollen.

```
10 REM..............................EINFACHES TELEFONVERZEICHNIS
20 PRINT "□◙ ◪      EINFACHES TELEFONVERZEICHNIS      "
30 PRINT "◙ WOLLEN SIE EIN NEUES VERZEICHNIS"
40 PRINT " ANLEGEN ◪ W ■ ,"
50 PRINT " EIN SCHON BESTEHENDES ERWEITERN ◪ A ■"
55 PRINT " ODER SICH EINE TELEFONNUMMER SUCHEN"
60 PRINT " LASSEN ◪ S ■ ?"
65 PRINT "◙◙ BITTE DIE ENTSPRECHENDE TASTE DRUECKEN!"
70 GET Z$
75 IF Z$="" OR (Z$<>"A" AND Z$<>"W" AND Z$<>"S") THEN GOTO 70
80 IF Z$="S" THEN GOTO 250
90 REM.........................................EINGABE DER DATEN
95 DIM N$(100),V$(100),VW$(100),RN(100)
100 PRINT "□◙ BEGINNEN SIE NUN MIT DER EINGABE."
105 PRINT "◙ (GEBEN SIE FUER DEN NAMEN UND VORNAMEN"
110 PRINT " '*,*' EIN, WIRD DER EINGABETEIL BEEN-"
115 PRINT " DET UND DIE DATEN WERDEN GESPEICHERT.)"
120 PRINT
125 I=0
130 PRINT: INPUT " NAME, VORNAME";N$(I),V$(I)
135 IF N$(I)="*" THEN GOTO 160
140 INPUT " VORWAHL,RUFNUMMER";VW$(I),RN(I)
150 I=I+1: GOTO 130
160 REM.................................ABSPEICHERN AUF DISKETTE
170 PRINT "◙◙ ◪ IST EINE DISKETTE EINGELEGT ? (J/N) "
180 GET D$: IF D$="" OR D$<>"J" THEN GOTO 180
190 OPEN 1,8,2,"TELEFON,S,"+Z$
200 FOR J=0 TO I-1
210 PRINT#1,N$(J)
215 PRINT#1,V$(J)
220 PRINT#1,VW$(J)
225 PRINT#1,RN(J)
230 NEXT J
```

```
235 CLOSE 1
240 END
250 REM...............................SUCHEN EINER TELEFONNUMMER
260 PRINT "▒▒ WESSEN TELEFONNUMMER SUCHEN SIE?"
270 PRINT: INPUT " NAME,VORNAME";GN$,GV$
280 PRINT "▒▒ ▒ IST DIE DISKETTE MIT DEM TELEFON-"
290 PRINT " ▒ VERZEICHNIS EINGELEGT? (J/N) "
300 GET D$: IF D$="" OR D$<>"J" THEN GOTO 300
310 OPEN1,8,2,"TELEFON,S,R"
320 INPUT#1,N$,V$,VW$,RN
330 IF N$=GN$ AND V$=GV$ THEN GOTO 360
340 IF ST=64 THEN GOTO 390
350 GOTO 320
360 CLOSE 1
370 PRINT "▒▒▒ DIE GESUCHTE TELEFONNUMMER IST:"
380 PRINT "▒ ";VW$;"/";RIGHT$(STR$(RN),LEN(STR$(RN))-1): GOTO 400
390 CLOSE 1: PRINT "▒▒ GESUCHTER NAME NICHT VORHANDEN!"
400 PRINT "▒▒ BRAUCHEN SIE NOCH EINE NUMMER? (J/N)"
410 GET W$: IF W$="" OR (W$<>"J" AND W$<>"N") THEN GOTO 410
420 IF W$="J" THEN GOTO 250
430 END

READY.
```

Zum besseren Verständnis folgt eine zeilenweise Programmerklärung:

Zeile 20 - 65
Bedienungsanleitung als Bildschirmausgabe.

Zeile 70 - 80
Hier verzweigt das Programm entsprechend Ihrer Wahl.

Zeile 95
In der DIM-Anweisung wird die maximale Zahl der Daten, die auf einmal eingegeben werden können, festgelegt.

Zeile 100 - 120
Erläuternder Text für die Eingabe.

Zeile 125 - 150
Falls die Bedingung zum Beenden der Eingabe nicht erfüllt ist, werden die Indizes erhöht.

Zeile 170 - 180
Vor dem Abspeichern muß natürlich eine Diskette im Laufwerk sein!

Zeile 190
Die Datei mit dem Namen "TELEFON" wird zum Schreiben oder Anfügen von Daten geöffnet.

Zeile 200 - 230
Die eingegebenen Daten werden auf die Diskette übertragen.

Zeile 235
Die Datei wird wieder geschlossen.

Zeile 270
Hier beginnt der Suchteil des Programms. Zunächst muß der Name des gewünschten Teilnehmers eingegeben werden.

Zeile 310 - 390
Die Datei wird geöffnet und die Daten werden gelesen. Falls der gesuchte Name gefunden wird, wird die Datei wieder geschlossen (Zeile 360) und die Telefonnummer ausgegeben (Zeile 370 - 380).

Wird der Name nicht gefunden, so wird überprüft, ob das Ende der Datei erreicht wurde (Zeile 340). Ist dies nicht der Fall, wird der nächste Datensatz gelesen und verglichen usw.

Zeile 400
Hier können Sie angeben, ob Sie noch weitere Telefonnummern suchen.

Die Variable ST aus Zeile 340 soll noch näher erläutert werden. Diese Variable besitzt eine festgelegte Funktion (genauso wie TI und TI$). Sie kann also nicht beliebig verwendet werden!

Der Commodore 64 benutzt die Variable ST für Eingabe- und Ausgabekontrollen. Der Wert von ST ändert sich z.B., wenn Fehler beim Kassetten- oder Diskettenbetrieb auftreten.

In unserem Beispiel wird ST gebraucht, um festzustellen, ob das Ende der Datei erreicht wurde: dies ist der Fall, wenn ST den Wert 64 annimmt.

Sollten Ihnen Einzelheiten dieses Programms noch unklar sein, arbeiten Sie am besten noch einmal die entsprechenden BASIC-Anweisungen durch.

5

Speicherbelegung

Einführung in die Speicherorganisation des Commodore 64

Der Commodore 64 besitzt eine Speicherkapazität von 64 K. Da 1 K = 1 Kilobyte = 1024 Speicherplätze bedeutet, sind das 64 * 1024 = 65536 Speicherplätze. In den weiteren Kapiteln werden wir häufig mit diesen Speicherplätzen zu tun haben. Deshalb werden wir uns im folgenden ausführlich mit der Organisation dieser Speicherplätze beschäftigen (s. auch Anhang).

Der Speicher eines Computers ist in mehrere Blöcke aufgeteilt, die verschiedene Funktionen besitzen.

D e r   R A M - S p e i c h e r

Der RAM-Speicher (engl. random access memory; d.h. Speicher mit wahlfreiem Zugriff) ist der sogenannte Schreib-Lese-Speicher. Hier werden Programme, Daten, Bildschirminhalte, Farbinformationen usw. gespeichert und können zur Verarbeitung wieder abgerufen werden.

All diese Informationen stehen jedoch nur solange zur Verfügung, wie der Computer in Betrieb ist, d.h. sie gehen beim Ausschalten verloren.

Der Commodore 64 besitzt verschiedene RAM-Speicher, die für die unterschiedlichsten Aufgaben gebraucht werden. Einer der für den Anwender wichtigsten Speicherbereiche ist der BASIC-Programm-Speicher, dessen Adressbereich von 2048 bis 40960 reicht. Der Benutzer hat also 38912 (= 40960 - 2048) Speicherplätze (38 K) für die BASIC-Programmierung zur Verfügung.

D e r   R O M - S p e i c h e r

Der ROM-Speicher (engl. read only memory; etwa: Speicher, aus dem nur gelesen werden kann) ist, im Gegensatz zum RAM-Speicher ein Permanentspeicher, d.h. daß der Speicherinhalt beim Ausschalten des Rechners erhalten bleibt.

Der ROM-Speicher kann nicht verändert werden. Somit ist es ausgeschlossen, daß der Anwender etwa versehentlich wichtige Speicherinhalte löscht, die den Rechner möglicherweise unprogrammierbar machen würden.

Der ROM-Speicher belegt die Adressen von 40960 bis 49152, besteht also aus 8 K. Er enthält die BASIC-Anweisungen des Commodore 64.

Die Anweisungen POKE und PEEK

Nachdem wir die verschiedenen Speicher des Computers kurz vorgestellt haben, soll im folgenden das Verfahren dargestellt werden, mit dessen Hilfe man bestimmte Werte direkt in einen Speicher schreiben kann. Dies geschieht mit der POKE-Anweisung. Sie lautet allgemein:

POKE

POKE A,B

Hierbei bezeichnet A einen Wert von 0 bis 65535, die Adresse eines Speicherplatzes. B bezeichnet einen Wert von 0 bis 255, der in den Speicher geschrieben werden soll.

Geben Sie nun die folgende Anweisung ein:

POKE 53281,7

Damit wurde in die Speicherzelle 53281, die die Hintergrundfarbe kontrolliert, der Wert 7 - für die Farbe Gelb - geschrieben.

Gewissermaßen das Gegenstück zur POKE-Anweisung stellt die Anweisung PEEK dar. Damit ist es möglich, den Inhalt einer bestimmten Speicheradresse abzufragen. Um zu erfahren, welcher Wert sich in der Speicherzelle A befindet, kann man die PEEK-Anweisung in Verbindung mit PRINT verwenden:

PEEK

PRINT PEEK(A)

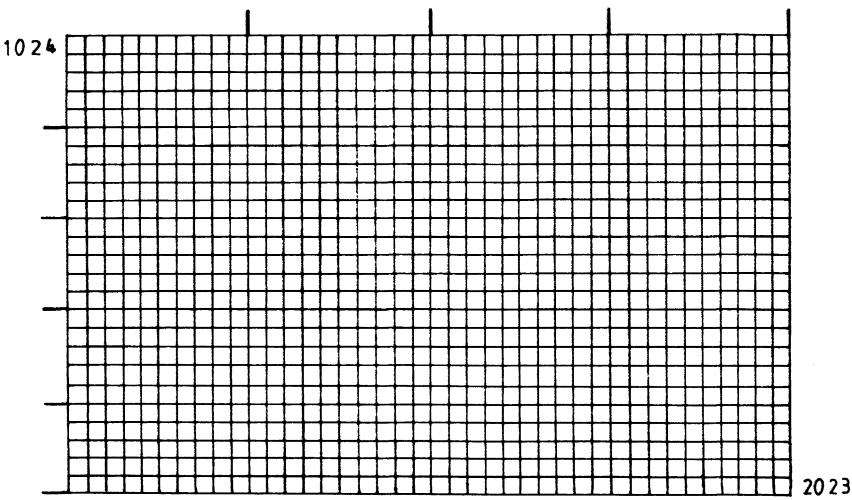
Das Ergebnis dieser Anweisung ist der augenblickliche Inhalt der Speicherzelle A, wobei A wieder eine Speicheradresse zwischen 0 und 65535 sein kann.

B i l d s c h i r m - u n d F a r b - s p e i c h e r

Der Bildschirmspeicher

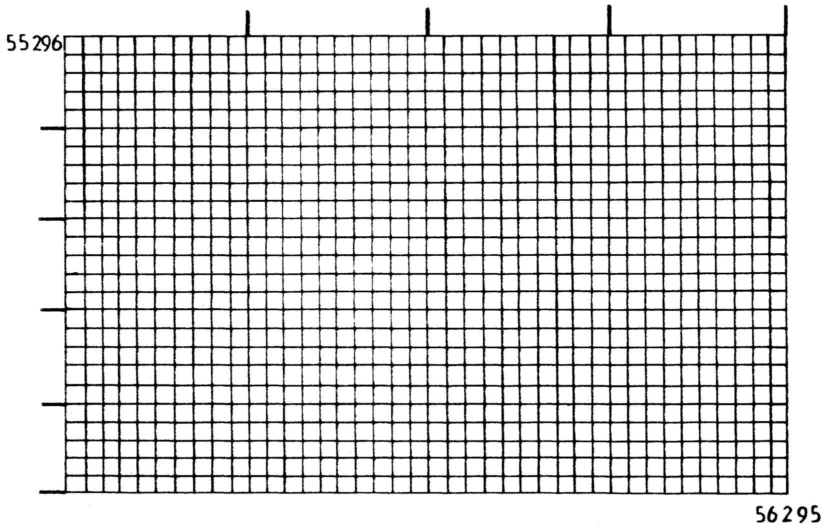
Der Bildschirmspeicher ist der Speicherbereich, in dem die Zeichen stehen, die gerade auf dem Bildschirm zu sehen sind.

Der Bildschirmspeicher beginnt bei der Adresse 1024 und kann insgesamt 1000 Zeichen beinhalten. Dies entspricht der Aufteilung des Bildschirms in 25 Zeilen zu je 40 Zeichen. Die folgende Abbildung verdeutlicht die Aufteilung dieses Speicherbereichs:



Der Farbspeicher

Der Adressbereich des Farbspeichers beginnt bei 55296. Ebenso wie der Bildschirmspeicher nimmt auch der Farbspeicher 1000 Speicherplätze ein: einen für jede Bildschirmposition. Die Aufteilung dieses Speichers ist der folgenden Abbildung zu entnehmen.



Zur Veranschaulichung der bisherigen Ausführungen soll das folgende Programm-Beispiel dienen:

```
10 REM..............................BILDSCHIRM UND FARBSPEICHER
20 PRINT"♡"
30 FOR I= 1024 TO 1063
40 POKE I,94
50 NEXT I
60 FOR I= 55296 TO 55335
70 POKE I,7
80 NEXT I

READY.
```

Mit diesem Programm wird die oberste Bildschirmreihe mit einem in gelber Farbe erscheinenden Grafikzeichen ausgefüllt.

Die Zeilen 30 - 40 geben an, welches Zeichen an welchen Bildschirmplätzen erscheinen soll. In unserem Beispiel soll also das Grafikzeichen mit der Code-Nr. 94 in die erste Bildschirmzeile - d.h. in die Speicherplätze von 1024 bis 1063 - geschrieben werden.

In den Programmzeilen 60 - 80 stehen die Adressen des Farbspeichers, die denen des Bildschirmspeichers entsprechen, also von 55296 bis 55335.

In Zeile 70 wird der Codewert für die gewünschte Farbe angegeben, also 7 für die Farbe Gelb.

Was passiert, wenn Sie die Zeilen 60 bis 80 weglassen? In dem Falle ist zwar das Zeichen in den Bildschirmspeicher gebracht worden; es erscheint jedoch in der Hintergrundfarbe und ist damit nicht sichtbar! Dies verdeutlicht, daß zu der Angabe, welches Zeichen erscheinen soll, auch die Angabe einer Farbe erforderlich ist.

Vordergrund- und Hintergrundfarbe

Neben der Möglichkeit, einzelne Zeichen farbig auszugeben, kann auch noch die Vorder- und Hintergrundfarbe gewählt werden.

Die Vordergrundfarbe kann durch das folgende kleine Programm angegeben werden:

```
10 REM....................................VORDERGRUNDFARBE
20 FOR I=0 TO 15
30 POKE 53281,I
40 FOR W=1 TO 500: NEXT W
50 NEXT I
```

Sie sehen der Reihe nach alle mit dem Commodore 64 darstellbaren Farben.

In Zeile 30 steht die Nummer des Speicherplatzes, der die Vordergrundfarbe bestimmt. Gibt man stattdessen die Adresse 53280 an, so werden die Hintergrundfarben dargestellt.

Damit haben wir zwei Speicherplätze kennengelernt, die schon interessante Farbmöglichkeiten bieten.

Der Mehrfarbenmodus

Zusätzlich zu der im vorletzten Abschnitt besprochenen Möglichkeit der Farbdarstellung gibt es die Möglichkeit, einzelne Zeichen auch mehrfarbig darzustellen. Dazu werden noch weitere Adressen benötigt, die diese Farben kontrollieren. Insgesamt kann man 4 Farben auswählen.

Probieren Sie dazu das folgende Programm aus:

```
10 REM...................................MEHRFARBENMODUS
20 PRINT"♥"
30 POKE 53281,15: REM...........................GRAU 3
40 POKE 53282,2: REM...............................ROT
50 POKE 53283,7: REM..............................GELB
60 POKE 53270,24: REM.............MEHRFARBENMODUS EINSCHALTEN
70 FOR Z=1 TO 20: PRINT"0 ";: NEXT Z
```

In den Zeilen 30 bis 50 stehen die jeweiligen Farben. Der in Zeile 30 angegebene Wert 15 für die Hintergrundfarbe steht für die Farbe Grau.

In den Zeilen 40 bzw. 50 ist die jeweilige Farbnummer für Rot und Gelb für die Buchstaben angegeben.

In Zeile 60 wird der Mehrfarbenmodus eingeschaltet.

Im nächsten Programmbeispiel lassen wir - durch Zufallszahlen bestimmte - farbige Bildschirmpunkte erscheinen:

```
10 REM.....................................FARBPUNKTE MIT CURSOR
20 PRINT "♥": POKE 53280,0: POKE 53281,0
30 FOR I=1 TO 1000
35 REM..................BERECHNUNG DES BILDSCHIRMSPEICHERPLATZES
40 A=INT(RND(1)*1000)+1024
45 REM......................................BERECHNUNG DER FARBE
60 F=INT(RND(1)*16)+1
80 IF PEEK(A)=160 THEN GOTO 110
90 POKE A,160: POKE A+54272,F
110 NEXT I
120 GET A$: IF A$="" THEN GOTO 120
130 PRINT "♥";: POKE 53280,14: POKE 53281,6
140 END

READY.
```

6

Grafik

Hochauflösende Grafik

Bisher hatten wir den Bildschirm in ein Raster eingeteilt, das aus 25 (Zeilen) mal 40 (Spalten) "Kästchen" bestand. An jede dieser 1000 Positionen konnte ein Zeichen gesetzt werden, z.B. ein Buchstabe oder ein Grafikzeichen.

In der 'Hochauflösenden Grafik' hingegen wird der Bildschirm in ein viel feineres Raster unterteilt, das aus 200 (Zeilen) mal 320 (Spalten) - insgesamt also 64000 - Punkten besteht, von denen jeder einzelne direkt angesprochen werden kann.

Geht man von der Standarddarstellung aus, so kann man sich den Bildschirm bei der hochauflösenden Grafikdarstellung auch so vorstellen, daß jedes der 1000 normalen Bildschirmkästchen noch einmal 'aufgelöst' wird in eine Matrix, die sich aus 8x8 Punkten zusammensetzt.

Für diese Grafikdarstellung besitzt der Commodore 64 einen speziellen Grafikbaustein, den sogenannten Video-Chip. Die für die Programmierung wichtigsten Adressen dieses Chips sollen im folgenden erläutert werden. Als erstes werden wir uns mit einer besonderen Darstellungsmöglichkeit innerhalb der hochauflösenden Grafik beschäftigen, mit der Programmierung von 'Sprites'.

(Ausführlichere Erläuterungen zu den grafischen Möglichkeiten dieses Computers finden Sie in unserem Buch "Grafik auf dem Commodore 64". Im einzelnen wird dort besprochen: die Verwendung von Grafik-Symbolen der Tastatur, Erstellung von (auch mehrfarbigen) Sprites (u.a. ein Hilfsprogramm zur Erstellung von Sprites auf dem Bildschirm) und die hochauflösende Grafik.)

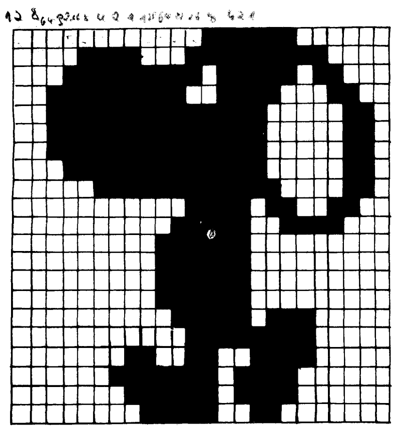
S p r i t e s

'Sprites' (dt.: Kobolde) sind vom Benutzer frei definierbare Figuren. Sie sind damit nicht mehr festgelegt durch die Möglichkeiten, die sich durch die Grafik-Zeichen der Tastatur bieten! Außerdem lassen sich Sprites, wenn sie einmal definiert sind, sehr einfach im ganzen Bildschirmbereich bewegen.

Allerdings gibt es zwei Einschränkungen: Jede Sprite-Figur besteht nur aus 512 Einzelpunkten, die sich aus einer Matrix von 24x21 Punkten zusammensetzen. Die zweite Einschränkung betrifft die Anzahl der Sprites, die gleichzeitig auf dem Bildschirm erscheinen können: es können höchstens 8 Sprites gleichzeitig auf dem Bildschirm erscheinen.

Die nächste Abbildung zeigt Ihnen das Schema einer Sprite-Figur. Wir wollen die Programmierung eines Sprites am Beispiel einer vielleicht nicht ganz unbekannten Figur erläutern. In dem folgenden Programm stehen die für das Sprite benötigten Daten in DATA-Zeilen.

SPRITE-ENTWURF



0	15	192
15	31	240
31	247	216
63	231	140
63	255	6
63	255	6
63	255	6
31	255	6
7	239	134
0	30	204
0	62	120
0	126	0
0	126	0
0	126	0
0	126	0
0	62	224
0	31	224
1	217	224
3	251	192
1	251	192
0	249	128

Wie berechnet man nun die Daten für eine Sprite-Figur? Zunächst einmal braucht man ein Raster in der entsprechenden Aufteilung, das also aus 24x21 Punkten besteht. In dieses Raster wird die gewünschte Figur eingezeichnet. Anschließend muß jedes Kästchen, das von der Zeichnung berührt wird, ausgefüllt werden.

Über die obere Reihe des Rasters wird dann von links nach rechts dreimal die Zahlenfolge 128, 64, 32, 16, 8, 4, 2, 1 geschrieben (s. auch Sprite-Datenblatt im Anhang).

Zur Illustrierung wollen wir einmal die Daten für die oberste Reihe der Sprite-Figur berechnen:

In der ersten Gruppe der ersten Zeile ist kein Kästchen ausgefüllt. Damit ist der dazugehörige Datenwert 0.

In der zweiten Gruppe sind die letzten vier Kästchen ausgefüllt. Addiert man nun die über den ausgefüllten Kästchen stehenden Zahlen, so erhält man den gesuchten Datenwert. In unserem Beispiel ergibt sich somit: 8+4+2+1 = 15.

Da in der dritten Gruppe die beiden ersten Kästchen ausgefüllt sind, erhält man als dritten Wert 192 durch Addition von 128 und 64.

Diese Berechnungen müssen für jede Sprite-Zeile durchgeführt werden, sodaß sich für ein Sprite insgesamt 3 x 21 = 63 Daten ergeben.

Als nächstes muß der Computer wissen, wo sich die Daten für ein Sprite befinden, und in welche Speicherplätze sie gebracht werden sollen, damit das Sprite an der gewünschten Stelle auf dem Bildschirm erscheint.

In unserem Programmbeispiel werden die Daten beginnend ab Speicherplatz 12288 abgelegt. Durch die Angabe POKE 2040,192 wird dem Computer mitgeteilt, wo die Daten für Sprite Nr. 0 stehen, wobei die Zahl 192 den Block bezeichnet, in dem die Daten stehen. Da jeder Block aus 64 Speicherplätzen besteht, gelangt man so zu der Anfangsadresse: 64 x 192 = 12288. Da kein größerer Wert als 255 in eine Speicherzelle geschrieben werden kann (s. Kapitel 9), ist die höchstmögliche Adresse für das Ablegen von Sprite-Daten gleich 255 * 64 = 16320.

Die Adresse 2040 gibt an, welches Sprite angesprochen wird. Von den 8 möglichen Sprites haben wir das Sprite Nr. 0 gewählt. Entsprechend wird durch die Adressen von 2041 bis 2047 festgelegt, an welcher Stelle die Daten für Sprite Nr. 1 bis Sprite Nr. 7 abgespeichert wurden.

Der letzte Schritt, um ein Sprite erscheinen zu lassen, ist noch die Angabe der Bildschirmposition, an der es stehen soll. Dazu werden die Adressen V+0 bis V+15 benötigt. Für jedes Sprite muß eine X- und eine Y-Koordinate angegeben werden. In unserem Beispiel ist die Figur 180 Punkte vom linken Bildschirmrand entfernt.

```
10 REM...............................SPRITE ERSCHEINEN LASSEN
19 REM.....................................BILDSCHIRM LOESCHEN
20 PRINT"□"
29 REM...........................SPRITE NR. 0 SOLL ERSCHEINEN
30 V=53248: POKE V+21,1
39 REM...........................................DATEN EINLESEN
40 FOR S1=12288 TO 12350: READ Q1: POKE S1,Q1: NEXT
49 REM............................ANGABE WO DIE DATEN STEHEN
50 POKE 2040,192
59 REM...............................FARBE FUER SPRITE NR. 0
60 POKE V+39,1
69 REM...............................ANGABE DER KOORDINATEN
70 POKE V,180: POKE V+1,180
99 REM.............................AB HIER STEHEN DIE DATEN
100 DATA 0,15,192,15,31,240,31,247,216
110 DATA 63,231,140,63,255,6,63
120 DATA 255,6,63,255,6,31,255,6,7,239,134,0,30,204,0,62
140 DATA 120,0,126,0,0,126,0,0,126,0,0,126
150 DATA 0,0,62,224,0,31,224,1,217,224
160 DATA 3,251,192,1,251,192,0,249,128

READY.
```

Zusammengefaßt lautet die Programmerklärung also:

Zeile 30	Der Video-Chip wird angesprochen, Sprite 0 soll erscheinen.
Zeile 40	Die Daten werden an die entsprechenden Adressen gebracht.
Zeile 50	Die Daten für Sprite Nr. 0 werden aus dem 192. Block ausgegeben.
Zeile 60	Die Farbe für das Sprite Nr. 0 wird bestimmt.
Zeile 70	Die Koordinaten für Sprite 0 werden angegeben.
Zeile 100 -160	Hier stehen die 63 Daten für die Sprite-Figur.

In unserem nächsten Beispiel wollen wir zeigen, wie eine Sprite-Figur auf dem Bildschirm bewegt werden kann. Zunächst aber noch einmal eine genauere Erklärung der benutzten Speicherplätze:

V + 21

Dieser Speicherplatz gibt an, welche Sprites auf dem Bildschirm erscheinen sollen. Wollen wir z.B. die drei Sprites mit den Nummern 2,3 und 5 auf den Bildschirm bringen, so müssen dazu die Zahlen 4 (= 2↑2), 8, (= 2↑3) und 32 (= 2↑5) addiert werden. Diese Summe, also 44, wird dann in den Speicher V+21 geschrieben, und bewirkt, daß die genannten drei Sprites gleichzeitig auf dem Bildschirm angezeigt werden: POKE V+21, 44.

V + 0 und V + 1

Hier werden die X- und Y-Koordinaten des ersten Sprites angegeben. Entsprechendes gilt für die übrigen Sprites für die Adressen von V+2 bis V+15, d.h. V+2 bzw. V+3 enthalten die X- bzw. Y-Koordinaten für das zweite Sprite usw.

2040 bis 2047

In diesen Speicherplätzen stehen die Anfangsadressen der Sprite-Daten.

V + 39 bis V + 46

Hier stehen die Farben der Sprites, wiederum mit Sprite 0 beginnend.

Mit dem letzten Programm können wir eine weitere Besonderheit der Sprites ausprobieren. Lassen Sie die Figur auf dem Bildschirm stehen und geben Sie die Anweisung

```
POKE V+23,1
```

ein. Als Ergebnis sehen Sie die Sprite-Figur in Y-Richtung vergrößert. Eine Vergrößerung in X-Richtung wird durch diese Anweisung erreicht:

```
POKE V+29,1
```

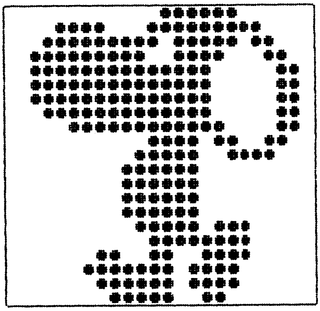
Soll das Sprite wieder in Originalgröße auf dem Bildschirm erscheinen, so müssen beide Speicher auf 0 gesetzt werden:

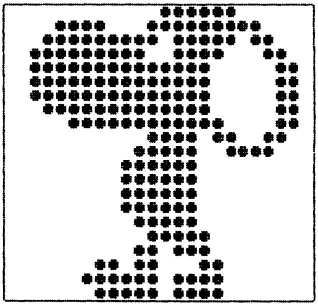
```
POKE V+23,0: POKE V+29,0
```

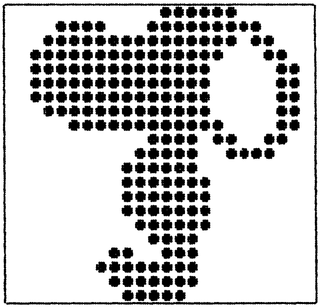
Im nächsten Programm wollen wir die Figur aus unserem Beispiel sich bewegen lassen.

Dazu müssen die Sprite-Entwürfe aus der nachfolgenden Abbildung ausgerechnet werden. Eine Bewegung entsteht dann dadurch, daß die einzelnen Figuren fortlaufend auf dem Bildschirm erscheinen, wieder gelöscht werden und etwas versetzt an der nächsten Stelle wieder erscheinen usw.

Auf die Art und Weise läßt sich ein kleiner Zeichentrickfilm zusammenbauen.







```
10 REM......................................BEWEGUNG VON SPRITES
270 REM.......................................BILDSCHIRM LOESCHEN
280 PRINT"□"
290 REM..............................................SPRITE NR. 0
300 V=53248
305 POKE V+21,1
310 REM.........................EINLESEN DER DATEN FUER 3 FIGUREN
320 FOR I1=12288 TO 12350: READ Q1: POKE I1,Q1: NEXT I1
330 FOR I2=12352 TO 12414: READ Q2: POKE I2,Q2: NEXT I2
340 FOR I3=12416 TO 12478: READ Q3: POKE I3,Q3: NEXT I3
350 REM.........................................FARBE SPRITE NR. 0
360 POKE V+39,1
370 P=192: REM...................................ANFANGSFIGUR
380 REM..............................................Y-KOORDINATE
390 POKE V+1,180
400 REM..............SCHLEIFE ZUR VERSCHIEBUNG AUF DEM BILDSCHIRM
410 FOR X=347 TO 0 STEP-3
420 R=INT(X/256)
430 POKE V,X-256*R: POKE V+16,R
440 REM...................ZUSAMMENSETZEN DER 3 FIGUREN ZU 'EINER'
450 POKE 2040,P
460 FOR T=1 TO 100: NEXT T
470 P=P+1: REM.................................NAECHSTE FIGUR
480 IF P>194 THEN P=192
490 NEXT X
500 GOTO 410
```

```
999 REM....................AB HIER STEHEN DIE DATEN
1000 REM....................1. FIGUR
1010 DATA 0,15,192,15,31,240,31,247,216,63,231,140,63,255,6,63
1020 DATA 255,6,63,255,6,31,255,6,7,239,134,0,30,204,0,62
1030 DATA 120,0,126,0,0,126,0,0,126,0,0,126
1035 DATA 0,0,62,224,0,31,224,1,217,224
1040 DATA 3,251,192,1,251,192,0,249,128
1100 REM...................2. FIGUR
1110 DATA 0,15,192,15,31,240,31,247,216,63,231,140,63,255,6,63
1120 DATA 255,6,63,255,6,31,255,6,7,239,134,0,30,204,0,62
1130 DATA 120,0,126,0,0,126,0,0,126,0,0,126
1135 DATA 0,0,62,0,0,31,0,0,59,128,1,177
1140 DATA 128,3,247,128,1,247,128
1200 REM...................3. FIGUR
1210 DATA 0,15,192,15,31,240,31,255,216,63,255,140,63,255,6,63
1220 DATA 255,6,63,255,6,31,255,6,7,239,134,0,30,204,0,62
1230 DATA 120,0,126,0,0,127,0,0,127,0,0,127,0,0,63,0
1240 DATA 0,30,0,0,206,0,1,254,0,0,254,0,0,124,0

READY.
```

Hier die Programmerklärung:

Zeile 300 - 305

Der Video-Chip wird angesprochen und Sprite Nr. 0 'eingeschaltet'

Zeile 320 - 340

Die Daten der jeweiligen Figuren werden nacheinander in verschiedene Blöcke abgelegt.

Zeile 360

Die Farbe (in diesem Fall weiß) wird für das Sprite Nr. 0 angegeben.

Zeile 390

Angabe der Y-Koordinate für das Sprite.

Zeile 410 bis 430

Bestimmung der X-Koordinaten. Die Angabe V+16 bewirkt, daß sich die Sprite-Figur von Position 256 bis zum rechten Bildschirmrand hin bewegen kann.

Zeile 450 bis 490

Die einzelnen Figuren werden nacheinander auf den Bildschirm gebracht. Damit die Bewegung nicht zu schnell abläuft, ist in Zeile 460 eine Warteschleife eingebaut.

Sprite - Übersicht

Die Anfangsadresse des Video-Chips lautet 53248 und wird der Variablen V zugewiesen.

V + 0 bis V + 15

Diese Speicheradressen kontrollieren die X- und Y-Koordinaten eines Sprites. Jeweils paarweise werden die Koordinaten eines Sprites zugeordnet. Die X- bzw. Y-Koordinate von Sprite 0 steht also in V+0 bzw. V+1 usw.

V + 16

Durch diesen Speicherplatz wird es ermöglicht, ein Sprite über die X-Koordinate 255 zu bringen, d.h. über den ganzen Bildschirm zu steuern. Dazu werden die den Sprites von 0 bis 7 entsprechenden Werte 1, 2, 4, 8, 16, 32, 64, 128 in den Speicher V+16 geschrieben.

V + 23

Diese Adresse gibt die Möglichkeit, jedes Sprite von 0 bis 7 in Y-Richtung zu vergrößern.

V + 28

Durch diesen Speicher wird der Mehrfarbenmodus eingeschaltet.

V + 29

Hier wird wie bei V+23 das Sprite vergrößert, allerdings in X-Richtung.

V + 30

In diesem Speicher wird eine Berührung von Sprites untereinander registriert.

V + 31

Gibt ebenfalls eine Berührung an, jedoch von Sprite und Hintergrund.

V + 37

Wird im Mehrfarbenmodus gebraucht und gibt die Farbe für den ersten gesetzten Punkt an.

V + 38

Ist ebenfalls für den Mehrfarbenmodus notwendig, und zwar für den zweiten gesetzten Punkt.

V + 39 bis V + 46

In diese Speicherplätze werden die Farben für die Sprites gesetzt, wenn die Sprites einfarbig erscheinen.

Hochauflösende Grafik

In diesem Abschnitt wollen wir beschreiben, wie man die hochauflösende Grafik des gesamten Bildschirms ausnutzen kann.

Der Commodore 64 besitzt leider keine speziellen Befehle zur Unterstützung der Grafik-Programmierung, sodaß sich die Programmierung von hochauflösenden Grafik-Darstellungen ähnlich aufwendig gestaltet wie die der Sprites.

Beginnen wir mit dem Einschalten der hochauflösenden Grafik! Die Anfangsadresse des Video-Chips für die Steuerung der hochauflösenden Grafik ist 53248. Diesen Wert weisen wir der Einfachheit halber wieder einer Variablen zu: V = 53248.

Durch die Anweisungen

```
POKE V+17,59 und
POKE V+24,24
```

wird auf den Grafikmodus umgeschaltet.

Zusätzlich kann nun noch eine Farbe angegeben werden, die in den Bildschirmspeicher übernommen wird. Die dazu notwendige Anweisungsfolge lautet:

```
FOR I = 1024 TO 2023: POKE I,HF+16*PF: NEXT I
```

Dabei ist HF der Wert der Hintergrundfarbe und PF der Wert der Punktfarbe.

Die Grafikdaten für einen vollständigen Grafikbildschirm stehen in den Speicherplätzen 8192 bis 16383 zur Verfügung.

Dieser Speicherbereich ist zunächst undefiniert (oder enthält noch alte grafische Daten) und muß deshalb vor dem Erstellen einer Grafik gelöscht werden. Dies kann durch die folgenden Anweisungen geschehen:

```
FOR I = 8192 TO 16383: POKE I,0: NEXT I
```

Dadurch wird in alle Speicherzellen des Grafikbereiches der Wert 0 geschrieben.

Mit dem folgenden Programm können Sie ein wohlgeformtes Herz auf dem Bildschirm darstellen.

Bemerkenswert daran ist wohl, daß eine einzige mathematische Funktion die herzförmige Kurve erzeugt. Diese Funktion lautet:

$$Y = \frac{2}{3}\left(\frac{X\uparrow 2 + |X| - 6}{X\uparrow 2 + |X| + 2} \pm \sqrt{36 - X\uparrow 2} \right)$$

```
10 REM.............................................HERZ-FUNKTION
20 REM........................................GRAFIK EINSCHALTEN
30 V=53248
40 POKE 53280,0: POKE V+17,59: POKE V+24,24
50 FOR I=1023 TO 2024: POKE I,32: NEXT I
60 FOR I=8192 TO 16383: POKE I,0: NEXT I
70 REM.................................................FUNKTION
80 FOR X=41 TO 279
90 I=(X-160)/20
100 Y1=2/3*((I↑2+ABS(I)-6)/(I↑2+ABS(I)+2)+SQR(36-I↑2))
110 Y2=2/3*((I↑2+ABS(I)-6)/(I↑2+ABS(I)+2)-SQR(36-I↑2))
120 Y1=-Y1*20+79
130 Y2=-Y2*20+79
140 FOR Y=Y1 TO Y2
150 GOSUB 200
160 NEXT Y
170 NEXT X
180 GET Z$: IF Z$="" THEN GOTO 180
190 POKE V+17,155: POKE V+24,21: PRINT"⊐": END
200 REM................................UNTERPROGRAMM PUNKT SETZEN
210 YK=320*INT(Y/8)+INT((Y/8-INT(Y/8))*8)
220 XK=8*INT(X/8)
230 EX=2↑(7-INT((X/8-INT(X/8))*8))
240 S=8192+YK+XK
250 POKE S,PEEK(S) OR EX
260 RETURN

READY.
```

Nun wieder die Programmerklärung:

Zeile 30
Die Anfangsadresse des Video-Chips wird einer Variablen zugewiesen.

Zeile 40
Die Rahmenfarbe wird auf schwarz eingestellt und die hochauflösende Grafik vorbereitet.

Zeile 50
Der Bildschirmspeicher kontrolliert in der hochauflösenden Grafik die Farbe. Hier wird die Farbe Rot für das Herz gewählt.

Zeile 60
Der Grafikspeicher wird vor dem Aufbau der Grafik gelöscht.

Zeile 80 bis 170
Das Bild wird aufgebaut.

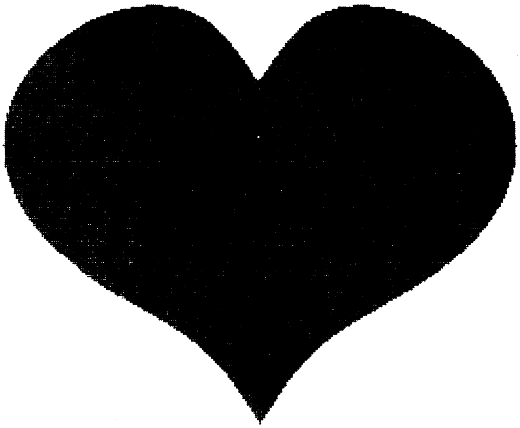
Zeile 80 bis 90
Die Grafik wird auf die Bildschirmgröße normiert.

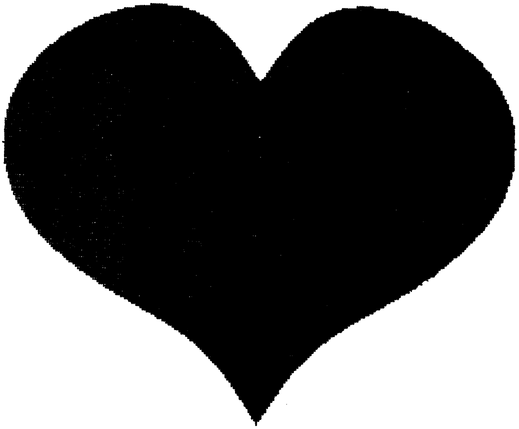
Zeile 100 und 110
Berechnung der Funktionsgleichung.

Zeile 120 bis 170
Berechnung der aktuellen Bildschirmpunkte.

Zeile 200 bis 260
Unterprogramm zur Berechnung der Grafikpunkte des Bildschirms.

Wir hoffen, daß Ihnen dieses Beispiel gefällt und machen noch einmal darauf aufmerksam, daß der Grafikaufbau sehr langsam geschieht: bis zur Fertigstellung dieser Grafik dauert es ca. 50 Minuten! - Falls Sie nicht soviel Geduld haben sollten, finden Sie nachstehend eine Abbildung der Herz-Grafik:





7

Musik

Musikprogrammierung

Der Commodore 64 besitzt - ebenso wie für die Grafik - auch einen speziellen Baustein zur Erzeugung und Programmierung von Musik. Dieser Chip wird kurz als SID bezeichnet, eine Abkürzung für 'Sound Interface Device'.

Mit diesem Musikbaustein kann man sowohl die unterschiedlichsten Geräuscheffekte - z.B. für Spiele - als auch eine Vielzahl von ganz verschiedenartigen Klängen erzeugen, wobei ein Musikstück bis zu drei Stimmen haben darf.

Im folgenden werden wir auf die Funktion der einzelnen Speicherplätze eingehen, die man zur Programmierung von Musik benötigt. Die Anfangsadresse des Musikchips weisen wir wieder einer Variablen - SI - zu. Sie liegt in diesem Fall bei 54272.

Grundlagen zur Musikprogrammierung

Die Erzeugung eines Tones hängt von einigen physikalischen Größen ab, die wir im einzelnen kennenlernen wollen. Zur Programmierung von Musikstücken genügt es erst einmal, zu wissen, daß ein Klang von verschiedenen Parametern bestimmt wird. Dazu gehören die Tonhöhe, Lautstärke, Wellenform und der Anschlag bzw. Toneinsatz.

Für das Programmieren einfacher einstimmiger Melodien genügt die folgende Übersicht der entsprechenden Speicherplätze:

1. Tonhöhe (SI u. SI + 1, bzw. 54272 u. 54273)

Die Tonhöhe wird durch die Frequenz bestimmt. Hierzu sind zwei Speicherplätze erforderlich.

2. Lautstärke (SI + 24 bzw. 54296)

Die Lautstärke kann in 16 Abstufungen eingestellt werden. Wird der Wert Null in diesen Speicherplatz geschrieben, so wird der Ton ausgeschaltet. Die Werte 1 bis 15 (maximale Lautstärke) geben dann die gewünschte Lautstärke an.

3. Wellenform (SI + 4 bzw. 54276)
Die Wellenform hat entscheidenden Einfluß auf einen Klang. Der Musikchip kann vier verschiedene Wellenformen erzeugen. Es sind Dreieck, Sägezahn, Rechteck und Rauschen wählbar.

4. Anschlag (SI + 5 u. SI + 6 bzw. 54277 u. 54278)
Durch diese beiden Speicherplätze läßt sich der Verlauf der Lautstärke eines Tones bestimmen. Dieser Verlauf wird durch die sogenannte Hüllkurve beeinflußt.

Im nächsten Programm wird eine einstimmige Melodie erzeugt, die einen ersten Eindruck von den musikalischen Fähigkeiten des Computers gibt.

```
10 REM.......................................EINFACHE MELODIE
20 SI=54272:REM.........................BASISADRESSE MUSIKCHIP
30 POKESI+5,9: POKESI+6,0:REM.....ANSCHLAG UND HALTEN(HUELLKURVE)
40 POKESI+24,15:REM.........................MAXIMALE LAUTSTAERKE
50 REM...................EINLESEN DER FREQUENZEN UND NOTENDAUER
60 READ HF,NF,NW
70 IF HF<0 THEN POKE SI+24,0: END
75 REM..................................EINLESEN DER FREQUENZEN
80 POKE SI,NF: POKE SI+1,HF
85 REM...................................WELLENFORM (SAEGEZAHN)
90 POKE SI+4,33
100 REM...........................................NOTENWERT
110 FOR I=1 TO NW
120 NEXT I
130 REM........................AUSSCHALTEN LAUTSTAERKE(RELEASE)
140 POKE SI+4,32
150 REM..........................................WARTESCHLEIFE
160 FORI= 1 TO 70 : NEXT
170 REM...........................RUECKSPRUNG FUER NAECHSTE NOTE
180 GOTO 60
```

```
199 REM......................FREQUENZEN IN REIHENFOLGE HF,NF,NW
200 DATA 34,192,128,39,32,128,43,224,256,52,32,256
210 DATA 52,32,256,58,128,256,52,32,256,43,224,256
220 DATA 34,192,384,39,32,128,43,224,256,43,224,256,39,32,256
230 DATA 34,192,256,39,32,768,34,192,128,39,32,128
240 DATA 43,224,256,52,32,256,52,32,256,58,128,256
250 DATA 52,32,256,43,224,256,34,192,256,39,32,256
260 DATA 43,224,256,43,224,256,39,32,256,39,32,256
270 DATA 34,192,768,-1,-1,-1

READY.
```

In diesem Beispiel wurden folgende Parameter verwendet:

In Zeile 20 wird die Anfangsadresse des Musikchips zugewiesen.

In Zeile 30 wird die Hüllkurve bestimmt, d.h. die Werte für den Anschlag und das Abschwellen werden angegeben.

In Zeile 40 wird die maximale Lautstärke eingestellt.

In Zeile 60 werden die jeweiligen Frequenzen und die Notenlänge eingelesen. Die beiden ersten Werte bestimmen die Frequenz, der jeweilige dritte Wert die Notendauer.

In Zeile 90 ist als Wellenform die Sägezahnform gewählt worden.

In Zeile 140 werden die Töne 'ausgeschaltet', d.h. die Lautstärke wird auf Null gesetzt.

Die Eingabe der Frequenzen erfordert noch einige Erläuterungen. Die Werte der Frequenzen (s. Anhang) für die erste Stimme werden in die Speicherplätze SI und SI+1 geschrieben. Zur Speicherung der Frequenzwerte werden zwei Speicherplätze benötigt, da in einem Speicher nur ein Wert bis 255 stehen kann, die Frequenzwerte jedoch viel höher sein können. Eine genaue Erklärung hierzu ist im Theorie-Kapitel zu finden. Die Frequenzen werden also in ein höherwertiges und ein niederwertiges Byte zerlegt. In unserem Beispiel werden dafür die Variablen HF und NF verwendet.

Wie schon erwähnt, bestimmt u.a. die Wellenform den Klangcharakter eines Tones. Im nächsten Beispiel wird das vorhergehende Programm als Unterprogramm eingesetzt und stellt nacheinander die Wellenformen Sägezahn, Dreieck und Rechteck vor. Diese akustische Demonstration wird Ihnen schnell deutlich machen, wie bestimmte Instrumente imitiert werden können.

```
10 REM.............................................WELLENFORMEN
20 SI=54272:REM..........................BASISADRESSE MUSIKCHIP
30 FOR I=SI TO SI+24: POKE I,0: NEXT I
40 REM.....................................1. FORM: SAEGEZAHN
45 PRINT"{clr}{down}{down}{down}{down}{down}{down}{down}{down}{down}{down}{down}{down}","{rvs on} SAEGEZAHN ": PRINT"{down}","   GEIGE"
50 POKE SI+24,15
60 WF=33: AN=5: AB=8: AS=5: AF=9
70 GOSUB 400
80 REM.................. .........................2. FORM: DREIECK
85 PRINT"{clr}{down}{down}{down}{down}{down}{down}{down}{down}{down}{down}{down}{down}","{rvs on} DREIECK ": PRINT"{down}","XYLOPHON"
90 RESTORE: WF=17: AN=0: AB=9: AS=0: AF=9
100 GOSUB 400
```

```
110 REM.........................................3. FORM: RECHTECK
115 PRINT"□▩▩▩▩▩▩▩▩▩▩▩","▨ RECHTECK ": PRINT"▩"," KLAVIER"
120 POKE SI+3,8: POKE SI+2,0: WF=65: AN=0: AB=9: AS=0: AF=0
130 RESTORE: GOSUB 400
140 POKE SI+24,0
150 PRINT"□"
160 END
400 REM....................................................MELODIE
410 POKE SI+5,16*AN+AB: POKE SI+6,16*AS+AF
420 READ HF,NF,NW
430 IF HF<0 THEN RETURN
440 POKE SI+1, HF: POKE SI,NF
450 POKE SI+4,WF
460 FOR I=1 TO NW: NEXT I
470 POKE SI+4,WF-1
480 REM..............................................NAECHSTE NOTE
490 GOTO 420
500 RESTORE: RETURN
599 REM........................DATEN IN DER REIHENFOLGE HF,NF,NW
600 DATA 34,192,128,39,32,128,43,224,256,52,32,256
610 DATA 52,32,256,58,128,256,52,32,256,43,224,256
620 DATA 34,192,384,39,32,128,43,224,256,43,224,256,39,32,256
630 DATA 34,192,256,39,32,768,34,192,128,39,32,128
640 DATA 43,224,256,52,32,256,52,32,256,58,128,256
650 DATA 52,32,256,43,224,256,34,192,256,39,32,256
660 DATA 43,224,256,43,224,256,39,32,256,39,32,256
670 DATA 34,192,768,-1,-1,-1

READY.
```

Der Unterschied zum vorhergehenden Programm besteht im wesentlichen darin, daß der Variablen WF nacheinander die Werte der jeweiligen Wellenform zugewiesen werden. Außerdem bekommen die Variablen AN(schlag), AB(schwellen), AS(Aushalten) und AF(Abfallen) verschiedene Werte.

Die Zeilen bis 140 bestimmen das Klangverhalten.

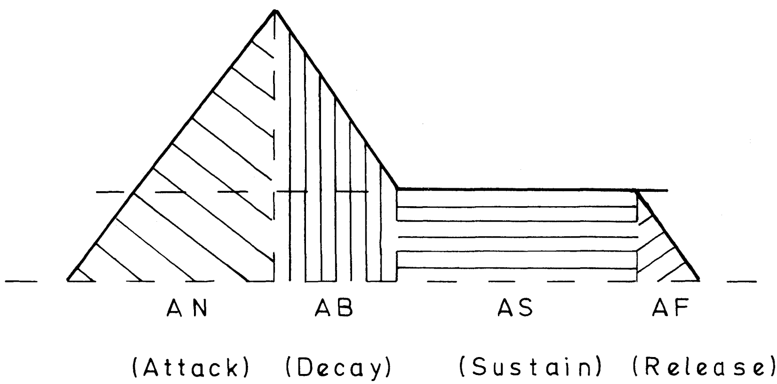
In der Zeile 410 erfolgt die Berechnung der Hüllkurve.

Die übrigen Programmzeilen werden Ihnen wohl bekannt vorkommen.

D i e H ü l l k u r v e

Zur genaueren Erklärung der Hüllkurve läßt sich am besten die folgende Abbildung heranziehen. Die entsprechenden englischen Bezeichnungen sind in Klammern angegeben.

HÜLLKURVE



Das Feld mit der Bezeichnung AN =Anschlag (Attack) stellt den Zeitraum dar, in dem ein Ton die maximale Lautstärke erreicht hat.

Das zweite Feld mit der Bezeichnung AB = Abschwellen (Decay) zeigt an, in welcher Zeit der Ton von seiner maximalen Lautstärke auf einen Haltepegel abfällt.

Das dritte Feld AS = Aushalten (Sustain) gibt die Zeitdauer des Aushaltens an.

Das vierte Feld AF = Abfallen (Release) gibt an, wann der Wert wieder auf Null abfällt.

Den Verlauf dieser Parameter gibt die eingezeichnete Hüllkurve an.

Im nächsten Beispiel stellen wir eine altenglische Melodie vor, die Heinrich der VIII. einer seiner Frauen gewidmet haben soll. Es wird eine Flöte nachgebildet, wobei verschiedene Hüllkurven ausprobiert werden. Wer die vollständige Melodie hören möchte, findet die entsprechenden Daten im Anschluß an dieses Programm.

```
10 REM.........................................HUELLKURVE
20 SI=54272: REM......................BASISADRESSE MUSIKCHIP
30 FOR I=SI TO SI+24: POKE I,0: NEXT I
40 REM.............. ....................WELLENFORM DREIECK
50 REM   POKE SI+4,17
60 POKE SI+24,15
70 PRINT"{CLR}{DOWN}{DOWN}{DOWN}{DOWN}{DOWN}{DOWN}{DOWN}{DOWN}{DOWN}{DOWN}{DOWN}","{RVS} DREIECK "
80 AN=2: AB=0: AS=10: AF=0
90 GOSUB 500
100 REM...............VERSCHIEDENEN HUELLKURVEN-EINSTELLUNGEN
110 AN=0
120 AB=0
130 AS=0
140 AF=0
150 IF (AN=0 AND AB=0 AND AS=0 AND AF=0) THEN GOTO 170
160 GOSUB 500
170 AF=15: GOSUB 500
180 IF AS=15 THEN GOTO 200
190 AS=15: GOTO 140
200 IF AB=15 THEN GOTO 220
210 AB=15: GOTO 130
220 IF AN=15 THEN GOTO 240
230 AN=15: GOTO 120
240 POKE SI+24,0: PRINT"{CLR}": END
```

```
500 REM..............................................MELODIE
510 PRINT"♥QQQQQQQQQQQQQ","ANSCHLAG =    ▮▮▮▮";AN
520 PRINT,"ABSCHWELLEN =    ▮▮▮▮";AB
530 PRINT,"AUSHALTEN =    ▮▮▮▮";AS
540 PRINT,"ABFALLEN =    ▮▮▮▮";AF
550 POKE SI+5,16*AN+AB
560 POKE SI+6,16*AS+AF
570 READ HF,NF,NW
580 IF HF<0 THEN RESTORE: RETURN
590 POKE SI+1,HF: POKE SI,NF: POKE SI+4,17
600 FOR I=1 TO NW/2: NEXT I
610 POKE SI+4,16
620 GOTO 570
630 RESTORE
640 RETURN
1000 REM......................DATEN IN DER REIHENFOLGE HF, NF, NW
1010 DATA 29,64,512,34,192,1024,39,32,512,43,224,768
1020 DATA 46,128,256,43,224,512,39,32,1024,32,224,512
1030 DATA 26,16,768,29,64,256,32,224,512,34,192,1024
1040 DATA 29,64,512,29,64,768,27,160,256,29,64,512
1050 DATA 32,224,1024,27,160,512,21,240,1024,-1,-1,-1

READY.
```

Greensleaves

```
105 REM...........................DATEN GREENSLEEVES IN NF,HF,NW
110 DATA 29,69,512,34,207,1024
120 DATA 39,18,512,43,219,768
130 DATA 49,58,256,43,219,512
140 DATA 39,18,1024,32,219,512
150 DATA 26,20,768,29,69,256
160 DATA 32,219,512,34,207,1024
170 DATA 29,69,512,29,69,768
180 DATA 27,160,256,29,69,512
190 DATA 32,219,1024,27,160,512
200 DATA 21,237,1024,29,69,512
210 DATA 34,207,1024,39,18,512
220 DATA 43,219,768,49,58,256
230 DATA 43,219,512,39,18,1024
240 DATA 32,219,512,26,20,768
250 DATA 29,69,256,32,219,512
260 DATA 34,207,768,32,219,256
270 DATA 29,69,512,27,160,768
280 DATA 24,157,256,27,160,512
290 DATA 29,69,1024,29,69,512
300 DATA 29,69,1536
310 DATA 52,39,1536,52,39,768
320 DATA 49,58,256,43,219,512
330 DATA 39,18,1024,32,219,512
340 DATA 26,20,768,29,69,256
350 DATA 32,219,512,34,207,1024
```
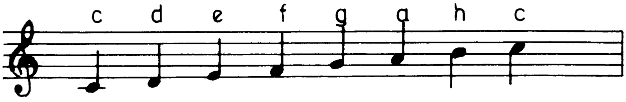
```
360 DATA 29,69,512,29,69,768
370 DATA 27,160,256,29,69,512
380 DATA 32,219,1024,27,160,512
390 DATA 21,237,1536,52,39,1536
400 DATA 52,39,768,49,58,256
410 DATA 43,219,512,39,18,1024
420 DATA 32,219,512,26,20,768
430 DATA 29,69,256,32,219,512
440 DATA 34,207,768,32,219,256
450 DATA 29,69,512,27,160,768
460 DATA 24,157,256,27,160,512
470 DATA 29,69,1536,29,69,1024
480 DATA -1,-1,-1

READY.
```

Die bisherige Noteneingabe läßt sich durch einige einfache Berechnungen noch vereinfachen. Die Angabe der Frequenz und des Notenwertes kann jeweils durch einen einzigen Wert erfolgen, aus dem sich der Computer selbst die anderen Parameter berechnen kann.

Zur Erinnerung an vielleicht vergessene Notenkenntnisse zeigt die nächste Abbildung die C-Dur-Tonleiter:

C-Dur Tonleiter



Eine Tonleiter besteht aus 12 Halbtonschritten, die folgendermaßen zueinander in Beziehung stehen: Die Frequenz eines Halbtons unterscheidet sich von der Frequenz des vorhergehenden Tones um den Faktor $\sqrt[12]{2}$, also 1.059463094. Dies ergibt sich aus der Tatsache, daß man das Intervall einer Oktave - also ein Frequenzverhältnis von 1:2 - in 12 Stufen mit gleichem Abstand unterteilt. Multipliziert man den angegebenen Faktor 12mal mit sich selbst, erhält man wieder die Zahl 2.

Man kann also jede Frequenz berechnen, wenn eine bestimmte Basis vorgegeben ist. Der Musikchip hat einen Umfang von acht Oktaven zur Verfügung. Für die Berechnung der gewünschten Töne geht man nun von einer Grundfrequenz je Ton aus, in unserem Fall von der ersten Oktave.

Das nächste Beispielprogramm stellt die Berechnungen im einzelnen vor. Die Noteneingabe erscheint zunächst etwas ungewöhnlich. Der entscheidende Vorteil liegt aber darin, daß die Datenmengen für ein Musikstück auf ein Drittel reduziert werden, da ja nicht mehr High- und Low-Byte sowie der Notenwert für jeden Ton einzeln angegeben werden müssen. Dieses Verfahren vereinfacht die Noteneingabe also tatsächlich ganz erheblich.

In Zeile 40 bis 60 wird die Grundfrequenz eingelesen; in unserem Fall die 1. Oktave.

Zeile 100 bestimmt den Notenwert, der hier in 1/16 Einheiten einzugeben ist.

In Zeile 140 wird der Ton für die gewünschte Oktave berechnet.

In den Zeilen 180 bis 200 werden Frequenzen und anschließend High- und Low-Bytes errechnet.
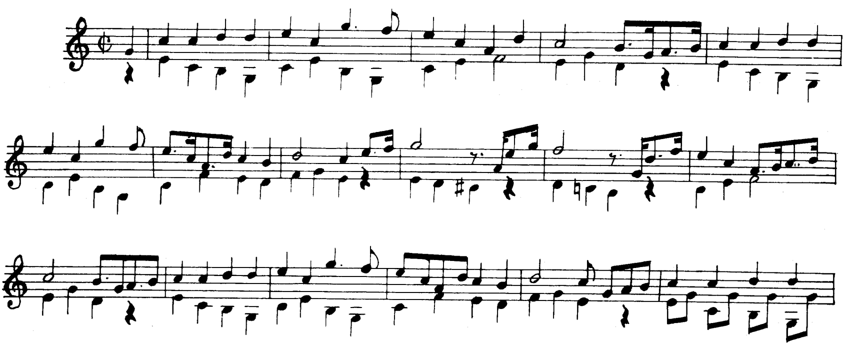
In Zeile 610 stehen die Daten für die Basisfrequenz.

```
10 REM.....................BERECHNEN DER NOTENDATEN AUS EINEM WERT
20 REM.................................EINLESEN DER GRUNDFREQUENZEN
30 DIM F(11)
40 FOR I=0 TO 11
50 READ F(I)
60 NEXT I
70 READ W: REM.....................Z.B.EINLESEN DER MELODIE-DATEN
80 IF W=0 THEN POKESI+24,0: END
90 REM.....................................................NOTENWERT
100 NW%= W/128
110 REM.......................................................OKTAVE
120 O%=(W-128*NW%)/16
130 REM.................................WELCHE NOTE IN DER OKTAVE
140 NR=W-128*NW%-16*O%
150 REM...................FREQUENZ DIESER NOTE IN DER GRUNDOKTAVE
160 F=F(NR)
170 REM.......................FREQUENZ FUER DIE BERECHNETE OKTAVE
180 F=F*2↑O%
190 REM..........................HIGH- UND LOW-BYTE DER FREQUENZ
200 HF%=F/256: NF%=F-256*HF%
230 GOSUB 500: REM.Z.B UNTERPROGRAMM.........ZUM SPIELEN DER NOTE
240 REM...........................................NAECHSTE NOTE
250 GOTO 70
600 REM.................................DATEN DER GRUNDFREQUENZEN
610 DATA 278,395,313,331,351,372,394,417,442,468,496,526
```

Auf die Möglichkeit des Musikchips zur Erzeugung mehrstimmiger Musikstücke wollen wir mit unseren nächsten Beispielen eingehen.

Dazu haben wir als erstes eine Sonatine ausgewählt. Die ersten Takte dieses Musikstückes sind hier angeführt. Das Stück wurde von einem italienischen Gitarristen komponiert, der ein Zeitgenosse von Ludwig van Beethoven war. Das Programm benutzt die ersten beiden Stimmen, die neben den bisher genannten Möglichkeiten auch noch durch Filter beeinflußt werden können.

Mauro Giuliani (1781-1829)
Sonatine Op. 71 No. 1



```
10 REM...................................ZWEISTIMMIGE MELODIE
30 SI=54272
40 FOR I=SI TO SI+24
50 POKE I,0
60 NEXT I
70 DIM HF%(1,2500),NF%(1,2500),WF%(1,2500)
80 DIM GF(11): REM....................ANZAHL DER GRUNDFREQUENZEN
90 W(0)=17: W(1)=33: REM.........WELLENFORMEN FUER STIMME 1 UND 2
100 POKE SI+22,8:   POKE SI+23,242: REM.....FILTER FUER STIMME 2
110 FOR I=0 TO 11: REM...............EINLESEN DER GRUNDFREQUENZEN
120 READ GF(I)
130 NEXT I
140 PRINT"{CLR}{DOWN}{DOWN}"
150 PRINT"{CLR}":PRINT" {RVS ON} GEDULD, ICH MUSS CA. 1 MIN RECHNEN "
160 TI$="000000"
170 REM.......................HIER BEGINNEN DIE DATENUMRECHNUNGEN
180 FOR K=0 TO 1: REM..............................ANZAHL STIMMEN
190 I=0
200 READ ND: REM..........................EINLESEN DER NOTENDATEN
210 PRINT"{HOME}{DOWN}{DOWN}{DOWN} ";MID$(TI$,4,1);" MIN UND ";RIGHT$(TI$,2);" SEK"
220 IF ND=0 THEN GOTO 400
230 WA=W(K): WE=WA-1: REM.................WELLENFORM-ANFANG/-ENDE
240 IF ND<0 THEN ND=-ND: WA=0: WE=0: REM.....ERKENNEN EINER PAUSE
250 NW%=ND/128: REM.....................BERECHNEN DES NOTENWERTES
260 O%=(ND-128*NW%)/16: REM..................BERECHNEN DER OKTAVE
270 REM.................STELLUNG DER AKTUELLEN NOTE IN DER OKTAVE
280 NT=ND-128*NW%-16*O%
290 FR=GF(NT): REM...............GRUNDFREQUENZ DER AKTUELLEN NOTE
300 FR=FR*2↑O%: REM.................AKTUELLE FREQUENZ DIESER NOTE
310 REM.......................HIGH- UND LOW-BYTE DIESER FREQUENZ
320 HF%=FR/256: NF%=FR-256*HF%
```

```
330 REM..........BESTIMMUNG DER TOENE IN SCHRITTEN VON 1/16-NOTEN
340 IFNW%=1THENHF%(K,I)=HF%:NF%(K,I)=NF%:WF%(K,I)=WA:I=I+1:GOTO20
350 FOR J=1 TO NW%-1
360 HF%(K,I)=HF%: NF%(K,I)=NF%: WF%(K,I)=WA: I=I+1
370 NEXT J
380 HF%(K,I)=HF%: NF%(K,I)=NF%: WF%(K,I)=WB
390 I=I+1: GOTO 200: REM.............................NAECHSTER TON
400 IF I>IMAX THEN IMAX=I: REM.....BESTIMMUNG DES HOECHSTEN INDEX
410 NEXT K: REM.....................................NAECHSTE STIMME
420 REM...............................ENDE DER BERECHNUNGEN!
430 POKE SI+5,0: POKE SI+6,240: REM......HUELLKURVE FUER STIMME 1
440 POKE SI+12,10: POKE SI+13,197: REM...HUELLKURVE FUER STIMME 2
450 POKE SI+1,HF%(0,I): POKE SI+8,HF%(1,I): REM........HIGH-BYTES
460 REM........................AB HIER WIRD DIE MELODIE GESPIELT
470 PRINT"▒▒ ▒  FERTIG!  "
480 PRINT"▒ MOECHTEN SIE DIE MELODIE JETZT HOEREN?  (J/N)"
490 GET Z$: IF Z$="" THEN GOTO 490
495 IF Z$="N" THEN PRINT"▒▒ SCHADE UM DIE ZEIT!": END
500 POKE SI+24,31
510 FOR I=0 TO IMAX
520 POKE SI,NF%(0,I): POKESI+7,NF%(1,I):REM LOW-BYTES DER STIMMEN
530 POKE SI+1,HF%(0,I): POKE SI+8,HF%(1,I): REM........HIGH-BYTES
540 POKE SI+4,WF%(0,I): POKE SI+11,WF%(1,I): REM.....WELLENFORMEN
550 FOR T=1 TO 40: NEXT T: REM.......................WARTESCHLEIFE
560 NEXT I: REM...............................NAECHSTE 1/16-NOTE
570 FOR T=1 TO 50: NEXT T: REM.......................WARTESCHLEIFE
580 POKE SI+24,0: REM............................LAUTSTAERKE  = 0
590 PRINT:INPUT" WIEDERHOLUNG (J/N)";W$
600 IF W$="J" THEN GOTO 500
```

```
610 END: REM......................................PROGRAMMENDE!
700 DATA 268,284,301,318,337,358,379,401,425,451,477,506
810 DATA 583,592,592,594,594,596,592,855,341,596
820 DATA 592,585,594,1104,459,199,457,203
830 DATA 592,592,594,594,596,592,855,341,468,208,457,210,592,587
840 DATA 1106,592,468,213,1111,-384,201,468
845 DATA 215,1109,-384,199,466,213,596,592
850 DATA 457,203,464,210,1104,459,199,457
855 DATA 203,592,592,594,594,596,592,855
860 DATA 341,468,208,457,210,592,587,1106,336,327,329,331,592
870 DATA 592,594,594,596,592,855,341,596,592,329,331,336,338
880 DATA 1104,331,327,329,331,592,592,594,594,596,592,855,341
890 DATA 340,336,329,338,592,587,1106,592,340,341,1111,-256
900 DATA 329,340,343,1109,-256,327,338,341,-256,340,338,336
910 DATA 331,329,341,338,1104,331,327,329,331,592,592,594,594
920 DATA 596,592,855,341,340,336,329,338,592,587,1106,592,0
1010 DATA 583,580,576,571,567,576,580,571,567,576
1020 DATA 580,1093,580,583,578,-512
1030 DATA 580,576,571,567,576,580,571,567,576,581,580,578
1040 DATA 581,583,580,-512,580,578,577,-512,578,576,571,-512
1050 DATA 576,580,1093,580,583,578,-512,580
1055 DATA 576,571,567,576,580,571,567,576
1060 DATA 581,580,578,581,583,580,-512,324,327,320,327,315,327
1070 DATA 311,327,320,327,324,327,315,327,567,320,327,324,327
1080 DATA 1077,311,327,324,327,578,-512,324,327,320,327,315,327
1090 DATA 311,327,320,327,324,327,315,327,567,576,581,324,327
1100 DATA 322,327,315,327,325,327,576,-512,324,325,324,322,577
1110 DATA -512,322,324,322,320,571,-512,1088,1077
1120 DATA 311,327,324,327,578,-512,324,327,320,327,315,327
1130 DATA 311,327,320,327,324,327,315,327,551,576,581,324,327
1140 DATA 322,327,315,327,325,327,576,-768,0

READY.
```

Wir hoffen, daß Ihnen die Melodie gefallen hat. Sollte es an einer Stelle allzu 'schräg' klingen, so überprüfen Sie am besten die DATA-Zeilen, da man sich beim Eingeben von DATA-Zeilen sehr leicht vertippen kann.

Das Programm wurde ausgiebig mit Kommentarzeilen versehen, die wohl alles wichtige erläutern.

Auf einen wesentlichen Unterschied zu den bisherigen einstimmigen Melodien wollen wir jedoch noch eingehen. Da beide Stimmen gleichzeitig erklingen müssen, ist eine vorhergehende Berechnung notwendig. Dabei werden die Töne in 1/16-Noten zerlegt und für jede einzelne 1/16-Note die Frequenzen (also LOW- und HIGH-Byte) und die jeweilige Wellenform in Feldern abgelegt. Dadurch entstehen während des Spielens keine unerwünschten Verzögerungen durch die Berechnungen, sodaß beide Stimmen gleichzeitig erklingen.

Die Dauer der Berechnungszeit wird im Programm angegeben. (Für diese und ähnliche Fälle bietet sich die Funktion TI$ an).

Mit diesem Programm können Sie leicht experimentieren, indem Sie z.B. Wellenform, Hüllkurve oder Filter verändern.

Das letzte Beispiel des Musik-Kapitels nutzt die Fähigkeit des Musikchips, auch dreistimmige Musikstücke zu spielen. Das Programm erzeugt die ersten Takte eines sicherlich bekannten Bourrees. Die restlichen Daten finden sich wieder nach dem Programm.

Dieses Programm unterscheidet sich von den vorhergehenden nur dadurch, daß zusätzlich die dritte Stimme zur Musikprogrammierung verwendet wird. Auch hier bietet es sich wieder an, die verschiedenen Parameter zur Musikerzeugung zu verändern.

```
10 REM......................................DREISTIMMIGE MELODIE
30 SI=54272
40 FOR I=SI TO SI+24
50 POKE I,0
60 NEXT I
70 DIM HF%(2,1000),NF%(2,1000),WF%(2,1000)
80 DIM GF(11): REM.....................ANZAHL DER GRUNDFREQUENZEN
90 W(0)=33: W(1)=65: W(2)=17: REM......WELLENFORMEN STIMMEN 1,2,3
100 POKE SI+10,8: POKE SI+22,128: POKE SI+23,240: REM......FILTER
110 FOR I=0 TO 11: REM...............EINLESEN DER GRUNDFREQUENZEN
120 READ GF(I)
130 NEXT I
140 PRINT"⊐⊠⊠"
150 PRINT"⊐":PRINT" ⊡ GEDULD, ICH MUSS CA. 1/2 MIN RECHNEN "
160 TI$="000000"
170 REM.......................HIER BEGINNEN DIE DATENUMRECHNUNGEN
180 FOR K=0 TO 2: REM..............................ANZAHL STIMMEN
190 I=0
200 READ ND: REM........................EINLESEN DER NOTENDATEN
210 PRINT"⊟⊠⊠⊠ ";MID$(TI$,4,1);" MIN UND ";RIGHT$(TI$,2);" SEK"
220 IF ND=0 THEN GOTO 400
230 WA=W(K): WE=WA-1: REM..................WELLENFORM-ANFANG/-ENDE
240 IF ND<0 THEN ND=-ND: WA=0: WE=0: REM.....ERKENNEN EINER PAUSE
250 NW%=ND/128: REM.....................BERECHNEN DES NOTENWERTES
260 O%=(ND-128*NW%)/16: REM..................BERECHNEN DER OKTAVE
270 REM.................STELLUNG DER AKTUELLEN NOTE IN DER OKTAVE
280 NT=ND-128*NW%-16*O%
290 FR=GF(NT): REM...............GRUNDFREQUENZ DER AKTUELLEN NOTE
300 FR=FR*2↑O%: REM..................AKTUELLE FREQUENZ DIESER NOTE
310 REM........................HIGH- UND LOW-BYTE DIESER FREQUENZ
320 HF%=FR/256: NF%=FR-256*HF%
```

```
330 REM..........BESTIMMUNG DER TOENE IN SCHRITTEN VON 1/16-NOTEN
340 IFNW%=1THENHF%(K,I)=HF%:NF%(K,I)=NF%:WF%(K,I)=WA:I=I+1:GOTO200
350 FOR J=1 TO NW%-1
360 HF%(K,I)=HF%: NF%(K,I)=NF%: WF%(K,I)=WA: I=I+1
370 NEXT J
380 HF%(K,I)=HF%: NF%(K,I)=NF%: WF%(K,I)=WB
390 I=I+1: GOTO 200: REM.............................NAECHSTER TON
400 IF I>IMAX THEN IMAX=I: REM.....BESTIMMUNG DES HOECHSTEN INDEX
410 NEXT K: REM...............................NAECHSTE STIMME
420 REM.................................ENDE DER BERECHNUNGEN!
430 POKE SI+5,9: POKE SI+6,10: REM.......HUELLKURVE FUER STIMME 1
440 POKE SI+12,85: POKE SI+13,133: REM...HUELLKURVE FUER STIMME 2
450 POKE SI+19,96: POKE SI+20,240: REM...HUELLKURVE FUER STIMME 3
460 REM.........................AB HIER WIRD DIE MELODIE GESPIELT
470 PRINT"□■ ■  FERTIG!  "
480 PRINT"■ MOECHTEN SIE DIE MELODIE JETZT HOEREN?  (J/N)"
490 GET Z$: IF Z$="" THEN GOTO 490
495 IF Z$="N" THEN PRINT"■■ DANN KOENNEN SIE MICH DURCHAUS......"
496 IF Z$="N" THEN FOR T=1 TO 2000: NEXT T
497 IF Z$="N" THEN PRINT"□■ ....ANDERWEITIG EINSETZEN!": END
500 POKE SI+24,31
510 FOR I=0 TO IMAX
515 REM.....................................LOW-BYTES DER STIMMEN
520 POKE SI,NF%(0,I): POKESI+7,NF%(1,I): POKE SI+14,NF%(2,I)
525 REM........HIGH-BYTES
530 POKE SI+1,HF%(0,I): POKE SI+8,HF%(1,I): POKE SI+15,HF%(2,I)
535 REM.....WELLENFORMEN
540 POKE SI+4,WF%(0,I): POKE SI+11,WF%(1,I): POKE SI+18,WF%(2,I)
550 FOR T=1 TO 25: NEXT T: REM.......................WARTESCHLEIFE
560 NEXT I: REM...............................NAECHSTE 1/16-NOTE
570 FOR T=1 TO 50: NEXT T: REM.......................WARTESCHLEIFE
580 POKE SI+24,0: REM............................LAUTSTAERKE  = 0
590 PRINT:INPUT" WIEDERHOLUNG (J/N)";W$
600 IF W$="J" THEN GOTO 500
```

```
610 END: REM.......................................PROGRAMMENDE!
700 DATA 268,284,301,318,337,358,379,401,425,451,477,506
800 REM.......................................DATEN 1. STIMME
810 DATA 340,342,599,342,340,595,340,342,587,337,339,596,338,336
820 DATA 587,329,327,582,327,329,331,329,327,326,580,340,342,599
830 DATA 342,340,595,340,342,587,337,339,596,338,336,587,329,327
840 DATA 582,-256,322,1095,0
900 REM........................................DATEN STIMME 2
910 DATA 311,310,564,569,571,569,567,566,564,566,567,569,571,569
920 DATA 567,571,308,310,311,310,564,569,571,569,567,566,564,566
930 DATA 567,569,578,578,1079,0
1000 REM.......................................DATEN STIMME 3
1010 DATA 356,358,615,358,356,611,356,358,603,353,355,612,354
1020 DATA 352,603,345,343,598,343,345,347,345,343,342,596,356
1030 DATA 358,615,358,356,611,356,358,603,353,355,612,354,352
1040 DATA 603,345,343,598,-256,338,1106,0

READY.
```

Bourrée

```
800 REM..........................................DATEN 1. STIMME
810 DATA 340,342,599,342,340,595,340,342,587,337,339,596,338,336
820 DATA 587,329,327,582,327,329,331,329,327,326,580,340,342,599
830 DATA 342,340,595,340,342,587,337,339,596,338,336,587,329,327
840 DATA 582,-256,322,1095
845 DATA 331,327,594,329,336,587,343,338,596,331,338,592,331
850 DATA 329,584,329,331,592,331,329,1609,338,329,587,343,338
855 DATA 596,331,338,592,345,340,598,337,340,594,337,331,331
860 DATA 330,-256,331,1611,347,342,600,342,340,601,340,343,598
865 DATA 340,338,599,338,341,596,345,340,598,337,340,595,331,-768
870 DATA 340,331,592,338,329,587,336,327,585,587,1095,-512,2119,0
```

```
900 REM..........................................DATEN STIMME 2
910 DATA 311,310,564,569,571,569,567,566,564,566,567,569,571,569
920 DATA 567,571,308,310,311,310,564,569,571,569,567,566,564,566
930 DATA 567,569,578,578,1079
940 DATA 567,566,562,567,571,576,568,569,578,580,569,580,569
945 DATA 313,315,313,311,566,578,567,571,576,568,569,577,578
950 DATA 570,571,580,582,566,315,314,315,321,579,571,580,578
955 DATA 577,569,578,576,571,567,576,571,578,566,827,320,315,313
960 DATA 568,313,324,582,311,323,580,310,321,323,326,564,313
965 DATA 324,571,569,571,571,1076,0
1000 REM.........................................DATEN STIMME 3
1010 DATA 356,358,615,358,356,611,356,358,603,353,355,612,354
1020 DATA 352,603,345,343,598,343,345,347,345,343,342,596,356
1030 DATA 358,615,358,356,611,356,358,603,353,355,612,354,352
1040 DATA 603,345,343,598,-256,338,1106
1045 DATA 347,343,610,345,352,603,359,354,612,347,354,608,347
1050 DATA 345,600,345,347,608,347,345,1625,354,345,603,359,354
1055 DATA 612,347,354,608,361,356,614,353,356,610,353,347,347
1060 DATA 346,-256,347,1627,363,358,616,358,356,617,356,359,614
1065 DATA 356,354,615,354,357,612,361,356,614,353,356,611,347,-768
1070 DATA 356,347,608,354,345,603,354,343,601,603,1111,-512,2135,0

READY.
```

Um einen umfassenden Überblick über die Speicherplätze des Musikchips zu bekommen, sind in der folgenden Übersicht noch einmal alle wichtigen Adressen kurz beschrieben. Im Anhang befinden sich weitere Tabellen dazu.

In der Programmsammlung befindet sich (wie schon erwähnt) ein Hilfsprogramm zum Musikmachen. (Ein für Grafik-Darstellungen sehr nützliches Hilfsprogramm zur Erstellung von Sprites findet sich in unserem Buch 'Grafik auf dem Commodore 64'). Solche Programmierhilfen erleichtern den Umgang mit den besonderen Möglichkeiten dieses Computers ganz erheblich.

Die musikalischen Fähigkeiten des Commodore 64 wurden nur beispielhaft aufgezeigt. Durch eigenes Programmieren werden Sie sicherlich noch viele weitere musikalische Ideen verwirklichen.

Übersicht über die Musikprogrammierung

Die Anfangsadresse des Musikchips ist SI = 54272.

1. Stimme

SI und SI+1
LOW- und HIGH-Byte für die Frequenzen werden hier abgelegt.

SI+2 und SI+3
Nur gültig bei der Rechteckschwingung: LOW- und HIGH-Byte der Pulsbreite.

SI+4
Die Wellenform wird festgelegt. Zur Erinnerung: 17 = Dreieck, 33 = Sägezahn, 65 = Rechteck, 129 = Rauschen.

SI+5
Anschlag (Attack) und Abschwellen (Decay).

SI+6
Aushalten (Sustain) und Abfallen (Release).

2. Stimme

SI+7 und SI +8
LOW- und HIGH-Byte der Frequenzen.

S+9 und SI+10
Pulsbreite (für Rechteckschwingung s.o. SI+2).

SI+11
Wellenform.

SI+12
Anschlag und Abschwellen.

SI+13
Aushalten und Abfallen.

3. Stimme

SI+14 und SI+15
LOW- und HIGH-Byte der entsprechenden Frequenzen.

SI+16 und SI+17
Pulsbreite (für Rechteckschwingung s.o. SI+2).

SI+18
Wellenform.

SI+19
Anschlag und Abschwellen.

SI+20
Aushalten und Abfallen.

Weitere wichtige Adressen des Musikchips:

SI+21 und SI+22
LOW- und HIGH-Byte der Filterfrequenz.

SI+23
Filterresonanz.

SI+24
Lautstärke und Filter.

SI+27
Rauschgenerator.

SI+28
Der Inhalt der Hüllkurve 3 kann ausgelesen werden.

8

Tips & Tricks

Einige Tips und Tricks

In diesem Kapitel sollen weitere Besonderheiten des Commodore 64 behandelt werden. Sie können sich ausführlich informieren über

- die Cursorsteuerung;
- die Möglichkeit, den Zeichensatz des Computers zu ändern, um eigene Zeichen zu definieren;
- das Verschieben des Bildschirmspeichers;
- das Verkleinern des BASIC-Speichers und
- die Benutzung der Funktionstasten.

C u r s o r s t e u e r u n g

Wollen Sie von einem Programm aus eine Ausgabe an eine ganz bestimmte Stelle des Bildschirms bringen, werden Sie feststellen, daß dazu die BASIC-Befehle des Commodore 64 nicht mehr ausreichen. Leider verfügt dieser Rechner nicht, wie die meisten anderen, über spezielle Anweisungen, mit denen man die gewünschte Zeile und Spalte für eine Bildschirm- bzw. Druckerausgabe angeben könnte.

Stattdessen besitzt der Commodore zwei verschiedene Arten der Cursorsteuerung. Die erste kennen Sie ja schon. Dabei handelt es sich um den normalen Modus, also die Bewegung des Cursors mit Hilfe der Cursorsteuertasten über den Bildschirm.

Sicherlich ist Ihnen auch schon aufgefallen, daß diese Art der Cursorbewegung nicht mehr funktioniert, wenn Sie vorher ein (bzw. eine ungerade Anzahl von) Anführungszeichen getippt haben. Dann erhalten Sie nämlich statt der gewünschten Cursorbewegung ein (inverses) Grafikzeichen. Jetzt befinden Sie sich im zweiten Modus der Cursorsteuerung. Hier wird die Bewegung nicht sofort ausgeführt, sondern sozusagen durch das Grafikzeichen codiert und erst dann ausgeführt, wenn die dazugehörige PRINT-Anweisung vom Programm ausgeführt wird.

Zu diesen Steuerzeichen gehören auch die HOME- und CLR-Taste. Sie ergeben - nach einem Anführungszeichen - ebenfalls einen Grafik-"Code".

Um diese Wirkungsweise zu verdeutlichen, probieren Sie doch einmal das folgende Beispiel aus:

```
10 REM......................CURSORTSTEUERUNG MIT PRINT-ANWEISUNG
20 PRINT"▒"
30 PRINT"▒▒▒▒▒ EINS";
40 PRINT"▒▒▒ ZWEI";
50 PRINT"▒▒ DREI";
60 PRINT"▒▒▒▒▒▒▒▒▒▒▒ VIER"
70 PRINT"▒▒▒▒▒": END

READY.
```

Der Nachteil hierbei ist, daß der Cursor an der Stelle stehenbleibt, die er nach der letzten PRINT-Anweisung hatte und beim nächsten PRINT-Befehl mit Steuerzeichen von dieser Stelle aus weiterbewegt wird, d.h. daß bestimmte Bildschirmpositionen nur relativ erreicht werden können.

Wollen Sie eine Textausgabe auf jeden Fall z.B. in der 5. Zeile beginnen lassen, unabhängig davon, wo sich der Cursor vorher befindet, bleibt Ihnen nichts anderes übrig, als entweder - durch PRINT" ▒ " - den Bildschirm zu löschen oder PRINT" ▒ " voranzuschicken. Danach steht auf jeden Fall der Cursor in der linken oberen Ecke des Bildschirms, sodaß Sie nun von da aus 5 Zeilen nach unten gehen können (durch **PRINT"▒▒▒▒▒"**).

Diesen Umstand kann man vermeiden, wenn man die gewünschte Cursorposition mit der POKE-Anweisung direkt in die entsprechenden Speicherzellen schreibt; darauf kommen wir gleich zu sprechen.

Zunächst sollte jedoch noch erwähnt werden, daß sich, ebenso wie Cursorbewegungen, auch Farbausgaben und Ein- bzw. Ausschalten der inversen Darstellung in einer PRINT-Anweisung "codieren" lassen. Wollen Sie z.B. das Wort "ROT" in rot auf dem Bildschirm erscheinen lassen, lautet die Anweisung dazu: **PRINT"▒ROT"**

Wollen Sie die gleiche Ausgabe invers darstellen, heißt die Anweisung:

PRINT"▓▓ ROT ■"

Anschließend finden Sie eine Liste mit den in diesem Buch verwendeten Cursor- und Farb-Steuerzeichen.

CURSOR-STEUERUNGS-SYMBOLE

```
PRINT"STEUERZEICHEN IN EINER PRINT-ANWEISUNG"

PRINT"▛      CLR-HOME-TASTE UND SHIFT
PRINT"       BILDSCHIRM WIRD GELOESCHT"
PRINT"▚      CLR-HOME-TASTE"
PRINT"       AUSDRUCK BEGINNT LINKS OBEN"

PRINT"▐      CURSOR-RECHTS-TASTE"
PRINT"       AUSDRUCK BEGINNT EINE SPALTE WEITER RECHTS"
PRINT"▌      CURSOR-LINKS-TASTE"
PRINT"       AUSDRUCK BEGINNT EINE SPALTE WEITER LINKS"
PRINT"▞      CURSOR-NACH-UNTEN-TASTE"
PRINT"       AUSDRUCK BEGINNT EINE ZEILE WEITER UNTEN"
PRINT"▜      CURSOR-NACH-OBEN-TASTE"
PRINT"       AUSDRUCK BEGINNT EINE ZEILE WEITER OBEN"

PRINT"▙      CTRL- UND 9-TASTE (ZUSAMMEN 'RVS ON')"
PRINT"       AUSDRUCK ERSCHEINT INVERS"
PRINT"▀      CTRL- UND 0-TASTE (ZUSAMMEN 'RVS OFF')"
PRINT"       INVERSER AUSDRUCK WIRD WIEDER ABGESCHALTET"


PRINT"FARBBEFEHLE IN EINEM PRINT-BEFEHL"

PRINT"◣      CTRL- UND 4-TASTE"
PRINT"       AUSDRUCK WIRD TUERKIS"
PRINT"▟      CTRL- UND 6-TASTE"
PRINT"       AUSDRUCK WIRD GRUEN"
PRINT"▄      CTRL- UND 7-TASTE"
PRINT"       AUSDRUCK WIRD BLAU"
PRINT"▖      CTRL- UND 8-TASTE"
PRINT"       AUSDRUCK WIRD GELB"
```

Bei der zweiten Möglichkeit, eine Ausgabe an einer bestimmten Bildschirmstelle zu erzeugen, brauchen Sie nicht mehr zu berücksichtigen, wo sich der Cursor befindet. Wir haben in einigen Programmen diese Möglichkeit in Form eines Unterprogramms benutzt. So kann man sich im Hauptprogramm auf die Angabe der gewünschten Zeile und Spalte beschränken.

Um das Wort "TEXT" z.B in der 6. Spalte der 10. Zeile beginnen zu lassen, reicht folgendes Programm:

```
10 REM.........................CURSORSTEUERUNG MIT POKE-BEFEHL
20 PRINT"□"
30 CZ=10: CS=6
40 GOSUB 1000
50 PRINT"TEXT"
60 END
1000 CZ=1024+40*CZ
1010 POKE 209,CZAND255: POKE 210,CZ/256: POKE 211,CS
1020 RETURN
```

Der Wert für CZ in Zeile 1000 entsteht auf folgende Weise: die Speicherplätze von 1024 bis 2023 beinhalten den Bildschirmspeicher. Durch 1024 + 40 * CZ wird also der Speicherplatz bestimmt, der für den Anfang der gewünschten Zeile zuständig ist. Anschließend wird dieser Wert für CZ in Low- und High-Byte zerlegt (das ist notwendig, da er größer ist als 255 - siehe auch Kapitel 9). Danach werden die so ermittelten Zahlenwerte mit POKE in die entsprechenden Speicherzellen geschrieben, die für die Zeilenposition des Cursors zuständig sind. Da der Wert für die Cursorspalte (CS) höchstens 39 ist, kann er ohne weitere Umrechnungen sofort in die zuständige Speicheradresse 211 gebracht werden.

Dieses Unterprogramm bietet sich dann an, wenn Sie innerhalb eines Programms mehrmals die Bildschirmposition bestimmen wollen, an der eine Ausgabe beginnen soll. Dagegen eignet sich die Methode der Steuerzeichen in PRINT-Anweisungen besonders dazu, Abstände zwischen

einzelnen Zeichen oder Wörtern bzw. Zahlen zu vergrößern oder zu verkleinern. Beide Methoden können natürlich auch nebeneinander angewandt werden. Sie können also z.B. mit Hilfe des Unterprogramms einen Text an einer bestimmten Stelle beginnen lassen und in der PRINT-Anweisung durch das entsprechende Steuerzeichen die Farbe bestimmen.

Änderung des Zeichensatzes

Der Commodore 64 bietet Ihnen die Möglichkeit, sich - zumindest für die Bildschirmausgabe - einen eigenen Zeichensatz zu definieren. Sie können einzelne Sonderzeichen festlegen, vollständige Zeichensätze für andere Sprachen (z.B. griechisch) oder auch Ihre eigene Geheimschrift entwickeln. Da der Standardzeichensatz des Rechners aus 256 Zeichen besteht, können Sie bei Bedarf ebenso viele Zeichen ändern und außerdem noch invers erscheinen lassen.

Allerdings befinden sich die Daten für den Standardzeichensatz im ROM-Bereich, sind also gegen direkte Änderungsversuche geschützt. Einige wenige BASIC-Anweisungen genügen aber, um die Werte des ROM-Bereichs in einen RAM-Bereich zu übertragen, wo dann beliebige Änderungen der Speicherinhalte möglich sind.

An einem kurzen Beispiel wollen wir Ihnen zunächst vorführen, wie ein neues Zeichen auf dem Bildschirm aussehen kann.

```
10 REM.........................DEFINITION EINES EIGENEN ZEICHENS
15 PRINT"□"
16 PRINT,"▪ BITTE WARTEN "
20 REM............................................VORBEREITUNG
25 POKE 55,0: POKE 56,48: CLR: REM......BASIC-SPEICHER BEGRENZEN
30 POKE 56334,0: REM.........................KEINE UNTERBRECHUNG
40 POKE 1,51: REM.........................ROM-BEREICH EINSCHALTEN
45 REM.........................................KOPIEREN STARTEN
50 FOR I=12288 TO 14335
60 POKE I,PEEK(53248+I-12288)
70 NEXT I
75 REM............................................KOPIEREN ENDE
80 POKE 1,55: REM.........ZURUECK ZUM NORMALEN VIDEO-CHIP BETRIEB
85 POKE 53272,28: REM...................NEUE ADRESSE ZEICHENSATZ
90 POKE 56334,1: REM................UNTERBRECHUNG WIEDER MOEGLICH
100 PRINT"□"
105 PRINT" A WIRD ZU ALPHA"
110 REM............................................'A' AENDERN
115 FOR I=12288+8 TO 12288+15
120 READ Q
130 POKE I,Q
140 NEXT I
145 REM............................................DATEN ALPHA
150 DATA 0,0,59,206,198,206,59,0

READY.
```

Wie Sie vielleicht schon erkannt haben, findet hier in Zeile 50 bis 70 das Kopieren des Zeichensatzes vom ROM- in den RAM-Bereich statt.

Um dieses Kopieren zu ermöglichen, sind jedoch noch einige Vorbereitungen nötig. Die Daten für den Zeichensatz liegen im ROM-Bereich an den Adressen 53248 bis 55295. Diese Speicheradressen kommen Ihnen vielleicht aus dem Grafik-Kapitel bekannt vor. Tatsächlich werden diese Speicher z.B. für Sprites und hochauflösende Grafik benutzt. Wieso können an denselben Stellen Daten für Buchstaben und Zahlen stehen? Die Erklärung dafür ist einfach.

Parallel zu dem Speicherbereich, der für die Grafik zuständig ist, gibt es - mit denselben Speicheradressen - einen weiteren ROM-Bereich, in dem die Zeichendaten stehen. Zu diesem Speicherbereich haben Sie als Benutzer normalerweise überhaupt keine Zugriffsmöglichkeit, auch nicht zum Lesen. Einzig und allein der Video-Chip, der ganz allgemein für die Darstellung von Zeichen, sei es Grafik, Sprites oder Buchstaben, zuständig ist, kann auf diesen ROM-Bereich zugreifen und "lesen", aus welchen Daten z.B. Buchstaben und Zahlen zusammenzusetzen sind.

Wollen Sie nun eigene Zeichen definieren, müssen Sie an die Speicher des Zeichengenerators herankommen. In diesem Fall wird der normale Betrieb des Video-Chips "aus"- und der ROM-Bereich des Zeichengenerators "eingeschaltet". Das geschieht durch die Anweisung: POKE 1,51. Da aber nach diesem Befehl der Video-Chip keine Kontrollmöglichkeiten über die Ein- und Ausgabe von Zeichen und rechnerinternen Programmunterbrechungen mehr hat (was zu seiner "normalen" Aufgabe gehört), dürfen Unterbrechungen - z.B. von der Tastatur her - nicht mehr möglich sein. Das wird - <u>vor</u> dem Einschalten des ROM-Bereichs - sichergestellt durch: POKE 56334,0.

Erst nach diesen Vorbereitungen können die Zeichensatzdaten in einen RAM-Bereich des Rechners kopiert werden.

Anschließend muß der normale Betrieb wiederhergestellt werden:

POKE 1,55	stellt den bereits bekannten Modus des Video-Chips wieder her, und
POKE 56334,1	ermöglicht wieder die Eingabe über die Tastatur.

Zusätzlich dürfen Sie nicht vergessen, dem Video-Chip mitzuteilen, daß er ab sofort die Daten für Bildschirmzeichen aus dem kopierten (und geänderten) RAM-Bereich holen muß, statt aus dem üblichen Zeichengenerator. Dazu dient die Anweisung: POKE 53272,28.

Falls Sie Ihren eigenen Zeichensatz an einer anderen Speicheradresse als 12288 (wie in unserem Beispiel) beginnen lassen wollen, geben wir Ihnen hier die verschiedenen Möglichkeiten an:

POKE-Wert für 53272	Anfangsadresse des Zeichensatzes
20	4096
22	6144
24	8192
26	10240
28	12288
30	14336

Anordnung der Zeichen im ROM-Bereich

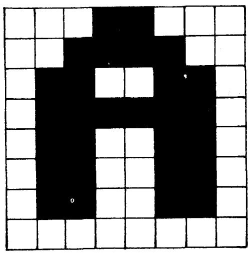
Anfangsadresse	Zeichen
53248	Großbuchstaben
53760	Commodore-Grafik-Zeichen
54272	Inverse Großbuchstaben
54784	Inverse Grafikzeichen
55296	Kleinbuchstaben
55808	Großbuchstaben und Grafikzeichen
56320	Inverse Kleinbuchstaben
56832	Inverse Großbuchstaben und Grafikzeichen

Damit sind (endlich!) alle Vorbereitungen abgeschlossen, und Sie können mit dem Verändern der Daten beginnen.

Jedes einzelne Zeichen besteht aus 8 Datenwerten. Die POKE-Werte der Zeichen (vgl. Commodore-Handbuch, S. 133 ff.) entsprechen der Stellung des Zeichens innerhalb eines Zeichensatzes. Um z.B. ein "A" darzustellen, werden also die Datenwerte von der 9. bis 15. Stelle benötigt.

Diese Datenwerte errechnen sich ähnlich wie z.B. Sprites-Daten. Jedes Zeichen wird in einem Raster von 8x8 Punkten dargestellt:

Wieder werden die Spalten von links nach rechts mit den Zahlen 128,64,32,16,8,4,2,1 versehen und je Zeile die Werte über einer Spalte aufaddiert. Für das "A" erhält man also die Datenwerte:



24
60
102
126
102
102
102
0

Um ein eigenes Zeichen zu definieren, müssen Sie es also auch zunächst in ein solches Raster zeichnen und die dazugehörigen Datenwerte ausrechnen. Dann können Sie die neuen Daten an die Stellen desjenigen alten Zeichens schreiben, das Sie durch das neue, selbstdefinierte ersetzen wollen. Das passiert im vorigen Beispiel in den Zeilen 110 bis 140.

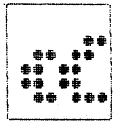
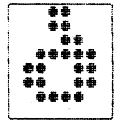
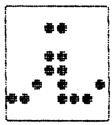
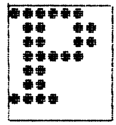
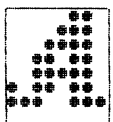
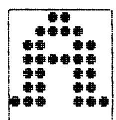
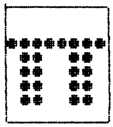
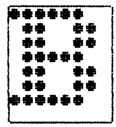
Das nächste Programm ermöglicht Ihnen die Erzeugung eines eigenen Zeichensatzes direkt am Bildschirm und ohne die umständliche Rechnerei.

```
50 REM..........................................ZEICHENGENERATOR
75 PRINT,"■■  ZEICHENGENERATOR "
100 PRINT"□": PRINT"■■ MIT DIESEM PROGRAMM KOENNEN SIE EINEN  ";
120 PRINT" EIGENEN ZEICHENSATZ ERSTELLEN.": PRINT
140 PRINT" NACH EINER KURZEN RECHENZEIT ERSCHEINT  EIN RAHMEN,";
160 PRINT" IN DEM DAS ZEICHEN GEAEN-   DERT WERDEN KANN.": PRINT
180 PRINT" DURCH DRUECKEN DER ENTSPRECHENDEN TASTE ";
200 PRINT"WIRD DER BUCHSTABE DARGESTELLT UND SIE";
220 PRINT"  KOENNEN AENDERUNGEN VORNEHMEN."
240 PRINT"■ DAS PROGRAMM IST BEENDET, WENN SIE"
260 PRINT" STATT EINES ZU AENDERNDEN ZEICHENS NUR"
280 PRINT" 'RETURN' DRUECKEN."
300 PRINT"■ (WEITER MIT TASTENDRUCK)"
310 GET A$: IF A$="" THEN GOTO 310
320 PRINT"□■ ■            ZEICHENGENERATOR          "
330 PRINT"■ BITTE ETWAS WARTEN!"
340 REM...................ZEICHENSATZ IN DEN RAM-BEREICH KOPIEREN
350 POKE 55,0: POKE 56,48: CLR
360 POKE 56334,0
370 POKE 1,51
380 FOR I=0 TO 2047:
390 POKE 12288+I,PEEK(55296+I)
400 NEXT I
410 POKE 1,55: POKE 53272,28 : POKE 56334,1
420 REM................DARSTELLUNG DER ZEICHEN AUF DEM BILDSCHIRM
430 PRINT"□"
440 REM...................................................RAHMEN
450 FOR I=1025 TO 1032
460 POKE I,100: POKE 54272+I,14
470 NEXTI
480 FOR I=0TO 7
490 POKE 1064+I*40,103: POKE 1073+I*40,101: POKE 55336+I*40,14
500 POKE 55345+I*40,14
510 NEXT I
```

```
520 FOR I=1385 TO 1392
530 POKE I,99: POKE I+54272,14
540 NEXT I
550 FOR I=0 TO 7
560 FOR J=0 TO 7
570 POKE 1065+I*40+J,32
580 NEXT J
590 PRINT"                    "
600 NEXT I
610 REM...................WELCHES ZEICHEN SOLL GEAENDERT WERDEN?
620 PRINT"                    "
630 PRINT"";
640 PRINT"WELCHES ZEICHEN SOLL GEAENDERT WERDEN?"
650 GET A$: IF A$="" THEN GOTO 650
660 REM..............................'RETURN' GEDRUECKT -> ENDE!
670 IF ASC(A$)=13 THEN GOTO 980
680 PRINT"                                        "
690 PRINT"";A$: A=PEEK(1024)
700 FOR I=0 TO 7
710 Z=PEEK(12288+8*A+I)
720 REM...................DARSTELLUNG DES ZU AENDERNDEN ZEICHENS
730 IF Z>=128 THEN POKE 1065+I*40,81: POKE 55337+I*40,14: Z=Z-128
740 IF Z>=64 THEN POKE 1066+I*40,81: POKE 55338+I*40,14: Z=Z-64
750 IF Z>=32 THEN POKE 1067+I*40,81: POKE 55339+I*40,14: Z=Z-32
760 IF Z>=16 THEN POKE 1068+I*40,81: POKE 55340+I*40,14: Z=Z-16
770 IF Z>=8 THEN POKE 1069+I*40,81: POKE 55341+I*40,14: Z=Z-8
780 IF Z>=4 THEN POKE 1070+I*40,81: POKE 55342+I*40,14: Z=Z-4
790 IF Z>=2 THEN POKE 1071+I*40,81: POKE 55343+I*40,14: Z=Z-2
800 IF Z>=1 THEN POKE 1072+I*40,81: POKE 55344+I*40,14: Z=Z-1
810 NEXT
```

```
820 PRINT"▌▌▌▌▌▌▌▌▌▌▌▌▌";: INPUT"BITTE AENDERN....";B$
830 EX=0
840 FOR I=0 TO 7
850 POKE 12288+B+8*A+I,0
860 NEXT I
870 FOR I=0 TO 7
880 AA=1065+I*40
890 FOR J=0 TO 7
900 IF PEEK(AA+J) <> 32 THEN EX=EX+2↑(7-J)
910 NEXT J
920 REM......................HIER WIRD DAS GEAENDERTE ZEICHEN GESPEICHERT
930 POKE 12288+8*A+I,EX
940 EX=0
950 NEXT I
960 REM....................ZURUECK ZUM ANFANG (NAECHSTES ZEICHEN)
970 GOTO 550
980 PRINT"▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒ DER NEUE ZEICHENSATZ IST ABGESCHLOSSEN!"
990 END
1000 END

READY.
```

Verschieben des Bildschirmspeichers

Wollen Sie auf Ihrem Commodore 64 Programme einsetzen, die für andere Commodore-Computer geschrieben worden sind, müssen Sie z.B. die POKE-Adressen angleichen.

Für die Bildschirmspeicherplätze bietet der Commodore 64 eine einfache Lösung: Der gesamte Bildschirmspeicher läßt sich an eine andere Adresse verschieben. Dazu werden nur zwei Speicherplätze benötigt. Der erste ist der Speicher 53272. Die oberen 4 Bits (das obere Nibble) sind zuständig für den Platz des Bildschirmspeichers. Die unteren Bits dieses Speichers bestimmen, welcher Zeichensatz benutzt wird. Wollen Sie also den Bildschirmspeicher verschieben, müssen Sie darauf achten, das untere Nibble der Adresse 53272 nicht zu beeinflussen! Das können Sie sicherstellen durch eine AND/OR-Verknüpfung (vergl. auch Kapitel 9).

Geben Sie dazu folgende Anweisung:

```
POKE 53272, (PEEK(53272) AND 15) OR X
```

Die möglichen Werte für X und die zugehörigen Adressen des Bildschirmspeichers entnehmen Sie bitte der Tabelle.

Als nächstes müssen Sie dem Speicher, der für die Bildschirmdarstellung zuständig ist, mitteilen, daß der Bildschirmspeicher verschoben wurde. Das geschieht mit der Anweisung

```
POKE 648,Y
```

Dabei ist Y die zu X gehörende Adresse (s. Tabelle) dividiert durch 256.

Anfang des Bildschirmspeichers bei	X	Y
1024	16	4
2048	32	8
3072	48	12
4096	64	16
5120	80	20
6144	96	24
7168	112	28
8192	128	32
9216	144	36
10240	160	40
11264	176	44
12288	192	48
13312	208	52
14336	224	56
15360	240	60

Ein Beispiel zum Ausprobieren:

Der Bildschirmspeicher soll bei Adresse 2048 beginnen. Also müssen folgende Anweisungen gegeben werden:

```
POKE 53272, (PEEK(53272) AND 15) OR 32
POKE 648,8
```

Daß der Bildschirmspeicher tatsächlich verschoben wurde, können Sie leicht überprüfen:

Nach den Befehlen:

```
POKE 2048,81    (neuer Anfang des Bildschirm-
                 speichers)
POKE 55296,7    (Anfang des Farbspeichers)
```

sehen Sie einen gelben Kreis in der linken oberen Ecke. Außerdem erkennen Sie hier, daß der Farbspeicher nicht verschoben wird; er bleibt immer an derselben Stelle.

Verkleinern des BASIC-Speichers

Wenn Sie die vorangegangenen Kapitel aufmerksam gelesen haben, ist Ihnen vielleicht aufgefallen, daß die Speicherplätze, die für die hochauflösende Grafik benutzt werden (8192 - 16383), mitten im BASIC-Speicher liegen (2048 - 40959), in dem sich auch Ihre Programme befinden.

Wenn Sie nun längere Programme schreiben, in denen Sie auch die hochauflösende Grafik benutzen, werden Sie feststellen, daß sich ein Programm, das mehr als 6 K Bytes Platz braucht, nicht mit der Grafik "verträgt".

Durch die für die Grafik notwendige Anweisung:

```
FOR I = 8192 TO 16383: POKE I,0: NEXT I
```

werden dann Programmteile gelöscht, die an diesen Stellen stehen.

Es gibt jedoch zwei Möglichkeiten, den BASIC-Speicher einzuschränken: Sie können die untere und/oder die obere Grenze selbst bestimmen. In dem Fall eines langen Programms mit hochauflösender Grafik bietet es sich an, den Anfang des BASIC-Speichers hinter den Grafik-Bereich, also auf 16384, zu legen. Dann haben Sie für Ihr Programm wesentlich mehr Platz zur Verfügung, ohne mit der Grafik in Konflikt zu geraten.

Die obere Grenze des BASIC-Speichers herunterzusetzen kann z.B. dann notwendig sein, wenn Sie Maschinenprogramme ablegen wollen, und der Platz im Kassettenpuffer oder in dem RAM-Bereich, der für Maschinenprogramme frei ist, nicht ausreicht. Legen Sie Maschinenprogramme im BASIC-Speicher ab, müssen Sie durch Heruntersetzen der BASIC-Grenze sicherstellen, daß diese Programme vom BASIC nicht berührt und verändert werden können.

Wie verändert man aber diese Grenzen?

Die Adresse für den Anfang des BASIC-Speichers steht in den Speicherplätzen 43 und 44, die Adresse für das Ende in den Plätzen 55 und 56.

Die Speicher 44 und 56 sind dabei jeweils für das höherwertige Byte (HB) zuständig, 43 und 55 für das niederwertige (NB) (vgl. auch Kapitel 9).

Die Berechnung geschieht wie folgt: Nehmen wir an, daß X die neue Anfangs- (oder End-) adresse für den BASIC-Speicher werden soll. Dann ist

HB = INT(X/256) und
NB = X - HB * 256

Um die Anfangsadresse zu verändern, brauchen Sie die Anweisungen:

```
POKE 44,HB: POKE 43,NB: POKE X-1,0: NEW
```

POKE X-1,0 ist notwendig, weil dadurch etwaige Maschinenroutinen, die im Bereich vor dem neuen BASIC-Speicher-Anfang liegen, einen Speicherplatz vor dem BASIC-Speicher zum Abbruch gezwungen werden und damit Ihr Programm nicht verändern können. Den NEW-Befehl brauchen Sie, um alle notwendigen Zeiger im Speicher richtig zu setzen.

Die obere BASIC-Speichergrenze wird verändert durch

```
POKE 55,NB: POKE 56,HB: CLR
```

Durch den CLR-Befehl werden alle etwa noch vorhandenen Variablen gelöscht. Lassen Sie ihn weg, erhalten Sie unter Umständen völlig falsche Variablenwerte in Ihrem Programm.

Benutzung der Funktionstasten

Die vier neben der normalen Tastatur befindlichen Funktionstasten können von Ihnen bis zu acht verschiedene Aufgaben zugewiesen bekommen, die dann während eines Programmlaufs auf Tastendruck ausgeführt werden.

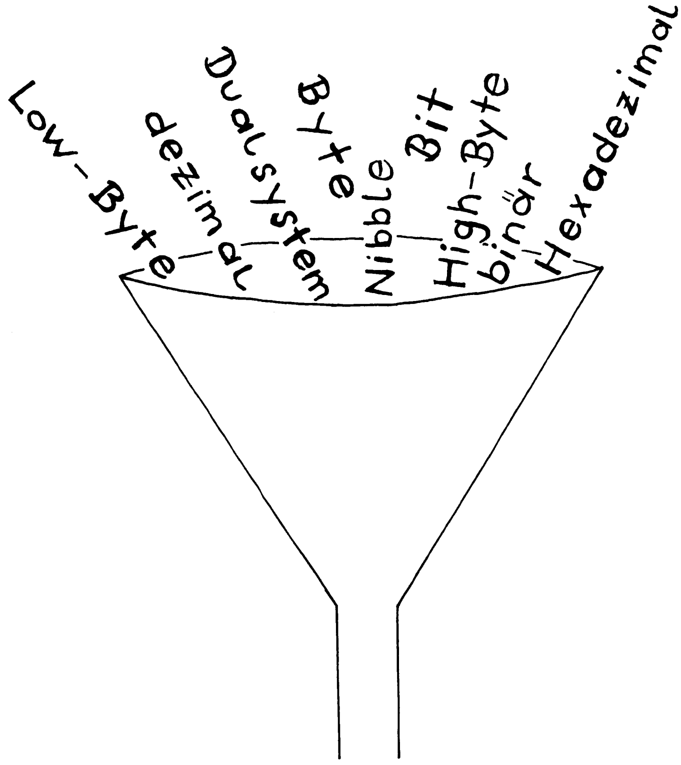
Das Prinzip ist einfach: Jeder Funktionstaste (F1 bis F8) entspricht ein ASC-Wert (der Tabelle im Commodore-Handbuch zu entnehmen). An geeigneter Stelle im Programm lassen Sie abfragen, ob z.B. die Taste F1 gedrückt wurde. Wenn dies der Fall ist, soll die entsprechende Funktion ausgeführt werden.

Im folgenden Beispiel werden die Zahlen von 1 bis 20 auf dem Bildschirm ausgegeben. Nach jeder Zahl wartet das Programm, bis eine Taste gedrückt wird. Ist diese Taste F1, wird hinter der Zahl deren Quadrat ausgegeben. Anschließend erwartet das Programm wieder einen Tastendruck. Drücken Sie nun F3, erhalten Sie noch die Wurzel aus der Zahl. Wählen Sie andere Tasten als F1 und F3, wird die nächste Zahl angezeigt.

```
10 REM...............................EINSATZ DER FUNKTIONSTASTEN
20 PRINT " X";TAB(10);"X*X";TAB(20);"SQR(X)"
30 FOR I=1 TO 20
40 PRINT I;
50 GET A$
60 IF A$="" THEN GOTO 50
70 IF ASC(A$)=133 THEN PRINT TAB(9);I*I;
80 GET Z$
90 IF Z$="" THEN GOTO 80
100 IF ASC(Z$)=134 THEN PRINT TAB(19);SQR(I);
110 PRINT
120 NEXT I
130 END

READY.
```

Natürlich können die Aufgaben, die durch die Funktionstasten F1 bis F8 ausgeführt werden sollen, auch weitaus komplexer sein. Komplizierte mathematische Funktionen können z.B. zu Beginn Ihres Programms durch DEF FN definiert werden. Oder aber Sie lassen Ihr Programm auf Tastendruck ganze Unterprogramme ausführen usw.


Low-Byte
dezimal
Dualsystem
Byte
Nibble
Bit
High-Byte
binär
Hexadezimal

9

Theorie

Etwas Theorie

B i t, B y t e u n d N i b b l e

Im Kapitel über die Speicheraufteilung des Computers haben Sie bereits erfahren, daß Sie vielen Speicherplätzen des Computers direkt bestimmte Werte zuweisen können (durch POKE). Von diesem BASIC-Befehl, der den angesprochenen Speicherplatz veranlaßt, seine spezielle Aufgabe auf eine ganz bestimmte Art und Weise auszuführen, wurde in den Grafik- und Musikkapiteln auch häufig Gebrauch gemacht. Um diesen Befehl jedoch effektiver anwenden zu können, sollten Sie noch genauer über die interne Organisation des Rechners Bescheid wissen.

Die Speicherplätze sind nämlich nicht die kleinsten Einheiten im Computer. Jeder Speicher ist noch einmal unterteilt in 8 Einheiten, die "bits" genannt werden. Diese Bits haben eine festgelegte Reihenfolge (Bit 0 bis Bit 7) und bilden zusammen ein "Byte". Das entspricht also wieder einem Speicherplatz des Rechners, den Sie sich damit folgendermaßen vorstellen können:

7 6 5 4 3 2 1 0   Bit-Nr.

Speicheradresse (z.B. 54272)

Außer Bit und Byte werden Sie vielleicht ab und zu noch auf einen dritten Fachausdruck stoßen: "Nibble". Ein Nibble bezeichnet die Hälfte eines Bytes.

Wir können also zusammenfassen: 4 Bits ergeben ein Nibble, 2 Nibble bilden ein Byte (= 1 Speicherplatz) und aus 65536 Bytes setzt sich der gesamte Speicher Ihres Commodore 64 zusammen.

Nun aber zurück zu den Bits:

Die einzelnen Bits eines Speicherplatzes können "gesetzt" oder "nicht gesetzt", bzw. "ein-" oder "ausgeschaltet" sein. Die verschiedenen Möglichkeiten, "gesetzte" und "nicht gesetzte" Bits einer Speicherzelle zu kombinieren, geben dem Computer an, wie im konkreten Fall die Aufgabe eines Speicherplatzes auszuführen ist.

Sie erinnern sich wahrscheinlich, daß genau zu diesem Zweck der POKE-Befehl benutzt wurde. Dennoch haben wir bei einer POKE-Anweisung dem Computer keineswegs gesagt: "In Speicher Nr. 54272 setze Bit 0, setze nicht Bit 1, setze Bit 2, ... ". Wo besteht da ein Zusammenhang? Mit den POKE-Befehlen werden Dezimalzahlen in die Speicher geschrieben. Diese Zahlen muß der Computer umrechnen, um zu wissen, welche Bits "ein"- bzw. "ausgeschaltet" sind, denn nur zwischen diesen beiden Zuständen von Bits kann er unterscheiden. Darum benutzt man zur Beschreibung dieser Vorgänge das duale (oder binäre) Zahlensystem.

Das Binär-System

Das duale oder binäre Zahlensystem benutzt nur die Ziffern 0 und 1 (das entspricht also genau den Zuständen eines Bits, die der Computer unterscheiden kann; dabei steht 0 für "aus-" und 1 für "eingeschaltet").

Genau wie im Dezimalsytem bestimmt auch im Dualsystem die Stelle innerhalb einer Zahl den Wert einer Ziffer (z.B. dezimal 296 = 1*6 + 10*9 + 100*2).

Im Dezimalsystem werden die Ziffern entsprechend ihrer Position in der Zahl mit Potenzen von 10 multipliziert (dabei geht man von rechts nach links vor!): 10↑0 = 1, 10↑1 = 10, 10↑2 = 100 usw.

Im Dualsystem werden stattdessen Potenzen von 2 verwendet: 2↑0 = 1, 2↑1 = 2, 2↑2 = 4, 2↑3 = 8 usw. Die Dualzahl 1011 wird also folgendermaßen in eine Dezimalzahl umgerechnet:

```
1011 = 1*2↑0 + 1*2↑1 + 0*2↑2 + 1*2↑3
     = 1*1   + 1*2   + 0*4   + 1*8
     = 11
```

Da ein Speicherplatz des Computers aus 8 Bits besteht, die alle den Wert 0 oder 1 haben können, ist der kleinste Wert, den ein Speicher aufnehmen kann:

00000000 (dual) = 0   (kein Bit gesetzt),

und der größte:

```
11111111 (dual) = 1*2↑0 + 1*2↑1 + 1*2↑2 + 1*2↑3
                + 1*2↑4 + 1*2↑5 + 1*2↑6 + 1*2↑7
                = 255   (alle Bits gesetzt).
```

Das ist also die Erklärung dafür, daß in keinen Speicher ein Wert geschrieben werden kann, der größer als 255 ist.

Außerdem können Sie sich nun das Sprite-Entwurfsblatt im Anhang als Gruppe von Bits vorstellen, die die Werte 0 oder 1 annehmen. Die Einteilung jeder Zeile in drei Gruppen zu je 8 Kästchen ist also deshalb notwendig, weil jedes Datenbyte aus genau 8 Bits besteht. Beim Addieren der Werte der ausgefüllten Kästchen haben Sie also schon Dualzahlen in Dezimalzahlen umgerechnet.

High- und Low-Byte

Was passiert nun aber, wenn Datenwerte größer als 255 sind, sich also nicht in einem Byte unterbringen lassen? In der Tabelle für die Notenwerte (im Anhang) z.B. beginnen die Frequenzen der Töne erst bei 268. In diesen Fällen werden zwei Speicherplätze benötigt. Der Datenwert (W) muß also in zwei Werte (High- und Low-Byte) zerlegt werden, die sich folgendermaßen berechnen:

```
High-Byte:  HB = INT(W/256)
Low-Byte:   LB = W - 256*HB
```

Das entspricht einer ganzzahligen Division mit Rest. Wollen Sie z.B. die Frequenz für C-5 (= 8583) in High- und Low-Byte zerlegen, rechnen Sie so:

8583/256 = 33 Rest 135

33 ist dann der Wert des High-Bytes und der Rest, 135, der Wert des Low-Bytes.

Diese Werte werden dann mit POKE entsprechend in die für High- und Low-Byte vorgesehenen Speicherstellen geschrieben.

Außerdem gibt es noch den Fall, daß nicht zwei Speicherstellen für eine Aufgabe, sondern ein Speicher gleich für zwei Aufgaben zuständig ist (jedes Nibble also eine andere Funktion hat). Das ist z.B. bei dem

Musikspeicher SI+24 (= 54296) der Fall, bei dem die Bits 0 bis 3 für die Lautstärke und die restlichen Bits für die Filtereinstellung sorgen. Aber auch bei den Sprites-Speichern, z.B. V+21, (= 53269) hat jedes Bit eine bestimmte Aufgabe. In diesen Fällen ist es notwendig, ganz bestimmte Bits "ein-" oder "auszuschalten". Das kann natürlich einmal dadurch geschehen, daß Sie die benötigten Dezimalzahlen ausrechnen. Wollen Sie aber, z.B. während eines Programmlaufs, in einer Speicherstelle ein Bit ändern, ohne die anderen zu beeinflussen, bietet sich eine andere Methode an, die Verknüpfung zweier Zahlen mit AND oder OR.

Bei beiden Verknüpfungen vergleicht der Computer die verknüpften Zahlen bitweise und errechnet aus dem Vergleich ein Ergebnis.

Sind zwei Zahlen durch AND verknüpft, wird ein Bit der Ergebniszahl nur dann "eingeschaltet" (=1), wenn beide entsprechenden Bits in den verknüpften Zahlen = 1 sind. Ist nur eines der beiden Bits der verknüpften Zahlen = 0, wird auch das Ergebnisbit = 0.

<u>Beispiel:</u> 141 AND 219

1 0 0 0 1 1 0 1	141
	AND
1 1 0 1 1 1 1 1	219
1 0 0 0 1 1 0 1	137

Die AND-Verknüpfung wirkt also wie ein Filter, der nur die Bits übrigläßt, die in den verknüpften Zahlen beide = 1 sind.

Die OR-Verknüpfung dagegen setzt im Ergebnis ein Bit = 1, wenn mindestens eins der entsprechenden Bits in den verknüpften Zahlen = 1 ist. Hier wird das Ergebnisbit nur dann 0, wenn alle entsprechenden Bits = 0 sind.

<u>Beispiel:</u> 141 OR 219

1 0 0 0 1 1 0 1	141
	OR
1 1 0 1 1 0 1 1	219
1 1 0 1 1 1 1 1	223

Entsprechend den unterschiedlichen Auswirkungen der AND- bzw. OR-Verknüpfung bieten sich in Verbindung mit dem POKE-Befehl zwei völlig verschiedene Einsatzmöglichkeiten an:

1. Sie wollen den Wert (den Sie aber vorher nicht berechnen können oder wollen) des Speichers X so ändern, daß Bit Nr. 2 auf jeden Fall "ausgeschaltet" ist.

Bit Nr. 2 (Dezimalwert = 4) können Sie "ausschalten" durch eine AND-Verknüpfung mit 251 (= 255 - 4). In 251 sind alle Bits gesetzt, außer eben Bit Nr. 2.

Die vollständige Anweisung lautet dann also:

POKE X, PEEK(X) AND 251.

2. Wollen Sie aber genau im Gegenteil sicherstellen, daß Bit Nr. 2 "eingeschaltet" ist, wählen Sie die OR-Verknüpfung und geben die Anweisung:

POKE X, PEEK(X) OR 4.

Dadurch behalten die ursprünglich gesetzten Bits ihre Werte (O oder 1) und Bit Nr. 2 wird auf jeden Fall gleich 1.

D a s   H e x a d e z i m a l - S y s t e m

Häufig wird in Programmierhandbüchern noch ein weiteres Zahlensystem verwendet, das Hexadezimalsystem. Im Commodore-Handbuch stehen z.B. in der Speicherbelegungstabelle als erste Angaben die Speicheradressen im Hexadezimalsystem (Hex-Code).

Dies ist ein Zahlen-System, das auf 16 (hexadezimal = 16) verschiedenen Zahlzeichen aufgebaut ist. Wie im Dezimal-System gibt es die Ziffern O bis 9. Hinzu kommen die Buchstaben A bis F, die als Zahlzeichen den (dezimalen) Werten von 10 bis 15 entsprechen. Wieder bestimmt die Stelle innerhalb einer Zahl den Wert der Ziffer, nur muß man jetzt mit Potenzen von 16 rechnen:

```
A90F = 15 * 16↑0 + 0 * 16↑1 + 9 * 16↑2 + 10 * 16↑3
       (F)                               (A)
     = 43279 (dezimal).
```

In diesem System lassen sich also mit nur 4 Stellen recht große Zahlen darstellen. Für die Dezimalzahlen von 0 bis 255, für die man ja im Dual-System bis zu 8 Stellen braucht, reichen hier zwei Stellen aus: 255 (dezimal) = FF (hexadezimal). Die größte mit 4 Stellen darstellbare Zahl im Hexadezimal-System (also: FFFF) ist übrigens 65535 - das entspricht 64K, der gesamten Speicherkapazität des Commodore 64. D.h. daß man mit einer 4-stelligen Hexadezimal-Zahl sämtliche Adressen (Speicherplätze) des Computers benennen kann.

Die folgende Tabelle enthält die Zahlen von 0 bis 32 in dezimaler, binärer und hexadezimaler Darstellung:

Dezimal	Binär	Hexadezimal
0	0	0
1	1	1
2	10	2
3	11	3
4	100	4
5	101	5
6	110	6
7	111	7
8	1000	8
9	1001	9
10	1010	A
11	1011	B
12	1100	C
13	1101	D
14	1110	E
15	1111	F
16	10000	10
17	10001	11
18	10010	12
19	10011	13
20	10100	14
21	10101	15
22	10110	16
23	10111	17
24	11000	18
25	11001	19
26	11010	1A
27	11011	1B
28	11100	1C
29	11101	1D
30	11110	1E
31	11111	1F
32	100000	20

10

Programmsammlung

Programmsammlung

D i g i t a l u h r

Eine einfache Möglicheit, die genaue Uhrzeit auf dem Bildschirm anzuzeigen, bietet die "eingebaute" Funktion TI$.

Im folgenden Programm sind die Ziffern für eine Digitaluhr aus Commodore-Zeichen zusammengesetzt und in DATA-Zeilen ablgelegt worden. Nach dem Programmstart werden diese Werte Feldvariablen zugewiesen.

Nun können Sie die Anfangszeit eingeben, die der Variablen TI$ zugewiesen wird. Das Programm läuft so ab, daß nach dem Vergleich mit dem jeweiligen Wert von TI$ an den entsprechenden Bildschirmplätzen die Ziffern der Digitaluhr ausgegeben werden.

```
10 REM...............................................DIGITALUHR
15 PRINT"{CLR}{DOWN} {RVS}                    DIGITALUHR           "
20 PRINT "{DOWN}{DOWN} BITTE EINEN AUGENBLICK WARTEN!"
25 DIM A%(9,34)
30 REM.......................................EINLESEN DER DATEN
40 FOR J=0 TO 9
50 FOR I=0 TO 34
60 READ Z%
70 A%(J,I)=Z%
80 NEXT I
90 NEXT J
100 REM.........................................STELLEN DER UHR
110 PRINT"{DOWN}{DOWN}{DOWN} ANFANGSZEIT IN DER FORM HHMMSS EINGEBEN"
120 INPUT Z$
125 IF LEN(Z$)>6 THEN PRINT "FALSCHE EINGABE!":GOTO 110
130 IF VAL(LEFT$(Z$,2))>24 THEN PRINT "FALSCHE EINGABE!":GOTO 110
135 TI$=Z$
140 PRINT"{CLR}"
150 POKE 53281,1: POKE 53280,1
160 REM...............................BILDSCHIRMSPEICHERPLAETZE
170 H1%=1385
180 H2%=1391
190 POKE 1637,224
200 M1%=1399
210 M2%=1405
220 POKE 1651,224
230 S1%=1413
240 S2%=1419
```

```
245 REM..........................................ANFANGSZEIT
250 F=VAL(LEFT$(TI$,1)): V=F: AW=H1%: H%=F: GOSUB 500
255 E=VAL(MID$(TI$,2,1)): V=E: AW=H2%: HH%=E: GOSUB 500
260 D=VAL(MID$(TI$,3,1)): V=D: AW=M1%: M%=D: GOSUB 500
265 C=VAL(MID$(TI$,4,1)): V=C: AW=M2%: MM%=C: GOSUB 500
270 B=VAL(MID$(TI$,5,1)): V=B: AW=S1%: S%=B: GOSUB 500
275 A=VAL(RIGHT$(TI$,1)): V=A: AW=S2%: SS%=A: GOSUB 500
280 REM.........................................LAUFENDE UHR
285 A=VAL(RIGHT$(TI$,1)): B=VAL(MID$(TI$,5,1))
290 C=VAL(MID$(TI$,4,1)): D=VAL(MID$(TI$,3,1))
295 E=VAL(MID$(TI$,2,1)): F=VAL(LEFT$(TI$,1))
300 IF SS%<>A THEN V=A: AW=S2%: SS%=A: GOSUB 500
305 IF S%<>B THEN V=B: AW=S1%: S%=B: GOSUB 500
310 IF MM%<>C THEN V=C: AW=M2%: MM%=C: GOSUB 500
320 IF M%<>D THEN V=D: AW=M1%: M%=D: GOSUB 500
330 IF HH%<>E THEN V=E: AW=H2%: HH%=E: GOSUB 500
340 IF H%<>F THEN V=F: AW=H1%: H%=F: GOSUB 500
345 GET Z$: IF Z$="E" THEN GOTO 360
350 GOTO 285
360 POKE 53280,14:POKE 53281,6:END
500 REM...................................UNTERPROGRAMM UHR
510 K=0
520 FOR I=0 TO 6
530 FOR J=0 TO 4
540 POKE AW+I*40+J,A%(V,K)
550 K=K+1
560 NEXT J
570 NEXT I
580 RETURN
```

```
999 REM...................................DATEN FUER ZIFFERN
1000 DATA 32,81,81,81,32,81,32,32,32,81,81,32,32,32,81
1001 DATA 81,32,32,32,81,81,32,32,32,81,81,32,32,32,81
1002 DATA 32,81,81,81,32
1010 DATA 32,32,81,32,32,32,81,81,32,32,81,32,81,32,32
1011 DATA 32,32,81,32,32,32,32,81,32,32,32,32,81,32,32
1012 DATA 81,81,81,81,81
1020 DATA 32,81,81,81,32,81,32,32,32,81,32,32,32,81,32
1021 DATA 32,32,32,81,32,32,32,81,32,32,32,81,32,32,32
1022 DATA 81,81,81,81,81
1030 DATA 81,81,81,81,81,32,32,32,81,32,32,32,81,32,32
1031 DATA 32,32,32,81,32,32,32,32,32,81,81,32,32,32,81
1032 DATA 32,81,81,81,32
1040 DATA 32,32,32,81,32,32,32,81,81,32,32,81,32,81,32
1041 DATA 81,32,32,81,32,81,81,81,81,81,32,32,32,81,32
1042 DATA 32,32,32,81,32
1050 DATA 81,81,81,81,81,81,32,32,32,32,81,81,81,81,32
1051 DATA 32,32,32,32,81,32,32,32,32,81,81,32,32,32,81
1052 DATA 32,81,81,81,32
1060 DATA 32,32,32,81,32,32,32,81,32,32,32,81,32,32,32
1061 DATA 81,81,81,81,32,81,32,32,32,81,81,32,32,32,81
1062 DATA 32,81,81,81,32
1070 DATA 81,81,81,81,81,32,32,32,81,32,32,32,32,81,32
1071 DATA 32,32,81,32,32,32,32,81,32,32,32,81,32,32,32
1072 DATA 32,81,32,32,32
1080 DATA 32,81,81,81,32,81,32,32,32,81,81,32,32,32,81
1081 DATA 32,81,81,81,32,81,32,32,32,81,81,32,32,32,81
1082 DATA 32,81,81,81,32
1090 DATA 32,81,81,81,32,81,32,32,32,81,81,32,32,32,81
1091 DATA 32,81,81,81,32,32,32,32,81,32,32,32,81,32,32
1092 DATA 32,81,32,32,32

READY.
```

08.43.17

17.35.08

W o c h e n t a g s b e s t i m m u n g

Ein Programm zur Wochentagsbestimmung kann z.B. bei der Programmierung eines Terminkalenders nützlich sein.

Das folgende Programm berechnet den Wochentag für ein beliebiges Datum zwischen 1901 und 2099. Die Bestimmung des Wochentages erfolgt dabei durch Zählen der Tage seit dem 1.1.1909. Für das Jahr J ergibt der Ausdruck

```
(J - 1901) * 1461 / 4
```

die Anzahl der Tage seit dem 1.1.1909 bis zum 31.12. des Vorjahres. Dazu muß noch die Tageszahl des laufenden Jahres hinzugezählt werden.

Die Monatsersten eines Nichtschaltjahres haben die Tagesnummern

0, 31, 59, 90, 120, 151, 181, 212, 243, 273, 304, 334

Diese Zahlenfolge kann man aus dem nachstehenden mathematischen Ausdruck erhalten:

```
INT((158 * M - 157/5) + (M + 1) * (M > 2)
```

Dabei ist M > 2 ein logischer Ausdruck, der im Commodore-BASIC den Wert -1 besitzt, wenn die angegebene Aussage ('M ist größer als 2') zutrifft, also wahr ist. Ist hingegen M nicht größer als 2, so ergibt sich der Wahrheitswert 0.

Bei Schaltjahren muß in den Monaten März bis Dezember die Tageszahl um jeweils eins erhöht werden. Dies läßt sich durch folgende Berechnung erreichen:

```
(INT(A/4) = A/4) * (M > 2)
```

Da der 1.1.1909 ein Dienstag war, kann nun durch Bestimmen des Siebenerrestes der gesuchte Wochentag ermittelt werden. Dazu müssen die Wochentage wie folgt codiert werden:

0 = Sonntag, 1 = Montag, 2 = Dienstag usw.

```
100 REM......................................WOCHENTAGSBESTIMMUNG
120 PRINT"▯▣ ▣          WOCHENTAGSBESTIMMUNG          ▣"
125 REM.....................................................EINGABE
130 PRINT"▣ DATUM IN DER FORM TT.MM.JJJJ EINGEBEN":PRINT:PRINT
140 INPUT" DATUM";D$
150 T=VAL(MID$(D$,1,2))
160 M=VAL(MID$(D$,4,2))
170 J=VAL(MID$(D$,7,4))
180 IFT>31 OR M>12 THEN PRINT" EINGABEFEHLER!":GOTO 140
190 IFJ<1901 THEN PRINT" EINGABEFEHLER!":GOTO   140
195 REM..................................................BERECHNUNG
200 GOSUB 300
210 D=D-7*INT(D/7)
220 FOR I=0 TO 6
230 READ W$(I)
240 NEXT I
250 REM.....................................................AUSGABE
260 PRINT"▣ DER ";MID$(STR$(T),2,2)"."MID$(STR$(M),2,2);
265 PRINT"."MID$(STR$(J),2,4);
270 PRINT" IST EIN ";W$(D);".":PRINT:PRINT:PRINT:RESTORE
280 PRINT:INPUT"WEITER J/N";A$
290 IFA$="J" THEN GOTO 120
295 END
300 REM ...................................TAGESNUMMER AB 1.1.1901
310 D=INT((J-1901)*1461/4)+1+T+INT((158*M-157)/5)
320 D=D+(M>2)*(M-(INT(J/4)<J/4))
330 RETURN
390 DATA SONNTAG,MONTAG,DIENSTAG,MITTWOCH
400 DATA DONNERSTAG,FREITAG,SAMSTAG

READY.
```

Numerische Integration

Die Berechnung eines bestimmten Integrals kann geometrisch als Berechnung einer Fläche betrachtet werden. In vielen Fällen ist eine Anwendung der üblichen Integrationsmethoden nicht möglich oder zu aufwendig, sodaß sich ein numerisches Verfahren anbietet. Da diese Methoden iterativ arbeiten, ist die Genauigkeit der Ergebnisse von der Zahl der Iterationen abhängig: je genauer das Integral berechnet werden soll, desto größer ist der dazu erforderliche Rechenaufwand.

Zu den gebräuchlichsten Verfahren zur numerischen Berechnung von Integralen gehören das Trapez-Verfahren, das Mittelpunkt-Verfahren sowie das Simpson-Verfahren.

Sowohl beim Trapez- als auch beim Mittelpunkts-Verfahren wird die Fläche unter einer Kurve durch eine Summe von Rechtecken approximiert: je schmaler die Rechtecke sind, desto genauer kann der Wert der Fläche bestimmt werden.

Bei der Trapez-Methode wird der Mittelwert der Funktionswerte der Punkte C und D benutzt, sodaß das entsprechende Flächenstück unter der Kurve als Trapez betrachtet wird.

Bei der Mittelpunkt-Methode wird der mittlere Funktionswert in dem Intervall von C bis D genommen, um ein Rechteck zu berechnen, das als Approximation für die Fläche unter der Kurve angesehen wird.

Bessere Ergebnisse als durch diese beiden Methoden erhält man in der Regel durch das Simpson-Verfahren, nach dem man einfach einen gewichteten Mittelwert aus den nach der Trapez- bzw. Mittelpunkt-Methode ermittelten Werten berechnet. Bezeichnen wir die Schätzung nach der Trapez-Methode mit T, die nach dem Mittelpunkt-Verfahren mit M und die nach dem Simpson-Verfahren mit S, so lautet die Simpson-Regel: S = 1/3 * T + 2/3 * M.

Zum Programm: die Funktions-Definition steht in Zeile 20, die Anfangs- und Endwerte des Integrals sind in der DATA-Anweisung in Zeile 200 angegeben. Zeile 210 enthält den Wert, der die Genauigkeit der Berechnungen festlegt. Dazu noch eine Anmerkung zu der Abfrage in Zeile 170: für eine aufwärts konkave Kurve ist der Wert

T zu groß, M hingegen zu klein. Das Gegenteil ist der Fall, wenn die Kurve abwärts konkav ist. In allen Fällen ist der Absolutbetrag der Differenz T-M größer als der tatsächliche Fehler, sodaß der Test in Zeile 170 ein brauchbares Abbruchkriterium liefert.

An vier Programmläufen sollen jetzt verschiedene Punkte illustriert werden. Dazu werden die folgenden vier Integrale berechnet:

(x↑2 + 3 * x + 1) dx

ln (x) dx

2 / (1 + x↑2) dx

e↑(-(x↑2)/2) dx

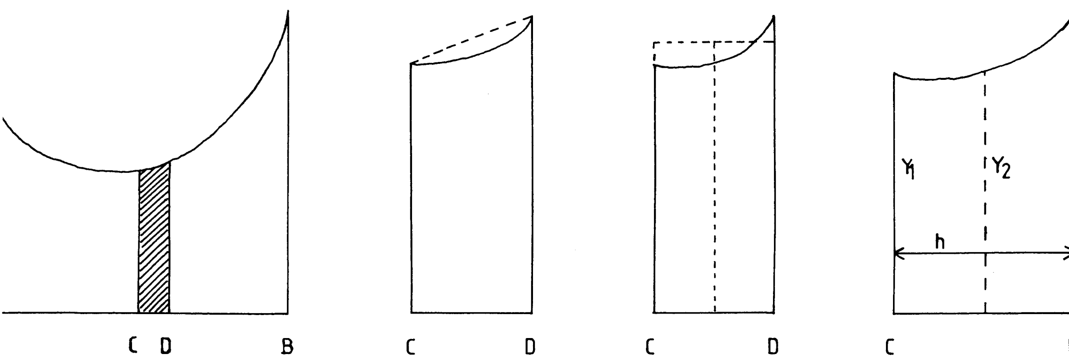
Das erste Integral ist aufwärts konkav, d.h. T ist zu groß und wird fortwährend vermindert, um den korrekten Wert zu erreichen. Der Anfangswert von M dagegen ist zu klein und wird entsprechend erhöht.

Die zweite Funktion ist abwärts konkav, sodaß das Gegenteil geschieht. In beiden Fällen kann man aber beobachten, daß die Schätzung nach Simpson sehr viel schneller zum Ziel führt als die beiden anderen (bei einem quadratischen Polynom sogar in einem einzigen Schritt!).

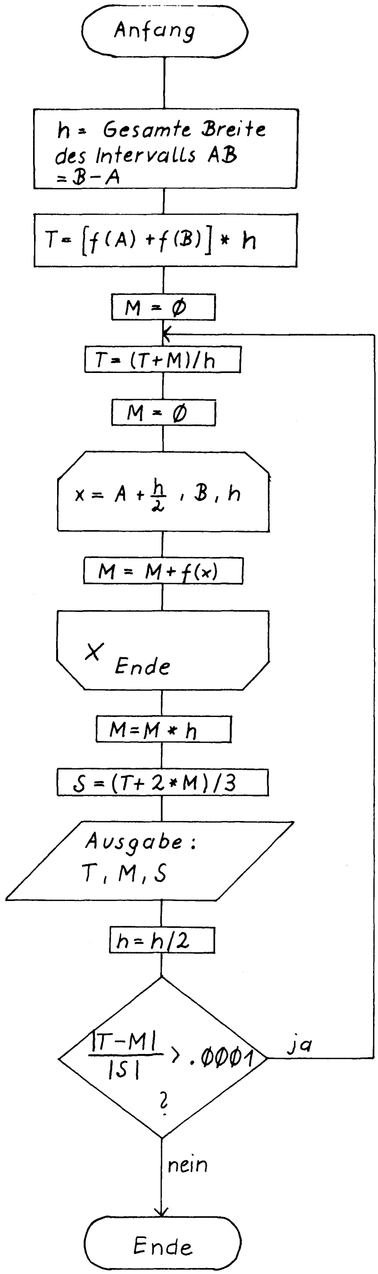
Der Wert des dritten Integrals ist gleich der Kreiszahl π.

Das vierte Beispiel schließlich ist eine Funktion, die überhaupt nur numerisch integriert werden kann.

NUMERISCHE-INTEGRATION



Trapez - Mittelpunkt-
Verfahren



Fluß-Diagramm: Integralberechnungen

```
5 REM........................................INTEGRALBERECHNUNG
10 PRINT"▢▣   ▩▥TRAPEZ-,▤MITTELPUNKT-,◣SIMPSON-VERFAHREN"
15 PRINT CHR$(154);
20 DEF FN F(X)=1/X: REM.......................ANGABE DER FUNKTION
30 READ A,B,E
35 REM..............................BERECHNUNG DER NAEHERUNGEN
40 H=B-A
50 T=(FNF(A)+FNF(B))*H
60 M=0
70 T=(T+M)/2
80 M=0
90 FOR X=A+H/2 TO B STEP H
100 M=M+FNF(X)
110 NEXT X
120 M=M*H
130 S=(T+2*M)/3
140 I=I+1
145 REM...............................AUSGABE DER NAEHERUNGEN
150 PRINT"▣";I;"▥";TAB(3);INT(T*100000000)/100000000;CHR$(154);
160 PRINT"▤";TAB(15);INT(M*100000000)/100000000;CHR$(154);
165 PRINT"◣"TAB(27);INT(S*100000000)/100000000;CHR$(154)
170 H=H/2
180 IF ABS(T-M)/ABS(S) > E THEN 70
195 REM................................................DATEN
200 DATA 1,10
210 DATA 1E-4

READY.
```

```
20 DEF FN F(X)=X↑2+3*X+1

200 DATA 1,5
```

```
  TRAPEZ-,MITTELPUNKT-,SIMPSON-VERFAHREN

 1  92          76          81.3333334
 2  84          80          81.3333334
 3  82          81          81.3333334
 4  81.5        81.25       81.3333334
 5  81.375      81.3125     81.3333334
 6  81.34375    81.328125   81.3333334
 7  81.3359375  81.3320313  81.3333334

READY.

20 DEF FN F(X)=LOG(X)

200 DATA 1,2
```

```
  TRAPEZ-,MITTELPUNKT-,SIMPSON-VERFAHREN

 1  .34657359   .4054651    .3858346
 2  .37601934   .39137966   .38625956
 3  .3836995    .38758831   .38629204
 4  .38564391   .38661936   .38629421
 5  .38613163   .3863757    .38629435
 6  .38625367   .3863147    .38629436
 7  .38628418   .38629944   .38629436

READY.
```

```
20 DEF FN F(X)=2/(1+X↑2)

200 DATA -1,1


  TRAPEZ-,MITTELPUNKT-,SIMPSON-VERFAHREN

 1  2           4           3.33333333
 2  3           3.2         3.13333333
 3  3.1         3.16235294  3.14156862
 4  3.13117647  3.14680051  3.1415925
 5  3.13898849  3.14289472  3.14159265
 6  3.14094161  3.14191817  3.14159265
 7  3.14142989  3.14167403  3.14159265

READY.

20 DEF FN F(X)=EXP(-(X↑2)/2)

200 DATA -4,4


  TRAPEZ-,MITTELPUNKT-,SIMPSON-VERFAHREN

 1  2.6837E-03  8           5.33422789
 2  4.00134185  1.08268226  2.05556879
 3  2.54201205  2.47055862  2.49437643
 4  2.50628534  2.50654758  2.50646017
 5  2.50641646  2.50649495  2.50646878

READY.
```

Division mit N Stellen

Falls Sie bei einer Division einmal mehr Stellen nach dem Komma berechnet haben wollen, als ihr Computer normalerweise liefert, besinnen Sie sich doch mal zurück, wie Sie früher solche Rechnungen "zu Fuß", auf einem Blatt Papier, ausgeführt haben! (Sie erinnern sich doch noch daran?)

Das angegebene Programm macht allerdings zusätzlich von einer Standardfunktion (INT) Gebrauch, die wir (Menschen) nicht "eingebaut" haben. Nach der Eingabe der zu teilenden Zahl (Zähler) und des Teilers (Nenner) wird die Zahl der Nachkommastellen erfragt. Der Teil des Ergebnisses, der vor dem Komma bzw. Dezimalpunkt steht (Q), wird mit sofort mit einer einzigen Anweisung berechnet und anschließend - mit einem Dezimalpunkt versehen - ausgegeben.

In einer FOR-Schleife wird nun jeweils der Rest der Division als neuer Zähler ermittelt und mit 10 multipliziert. Aus diesem neuen Teiler wird mit Hilfe der INT-Funktion dann die nächste Nachkommastelle berechnet und ausgegeben, usw.

```
10 REM...................................DIVISION MIT N STELLEN
20 PRINT"▒▒ ▒         DIVISION MIT N STELLEN        "
30 PRINT: INPUT" ZAEHLER";Z
40 PRINT: INPUT" NENNER";N
50 PRINT:PRINT:INPUT" ZAHL DER NACHKOMMASTELLEN";K
60 REM...................................BERECHNUNG UND AUSGABE
70 Q=INT(Z/N)
80 PRINT: PRINT: PRINT
90 PRINT Z;"/";N;"=";STR$(Q);".";
100 FOR I=1 TO K
110 REST=Z-(N*Q)
120 Z=REST*10
130 Q=INT(Z/N)
140 PRINT RIGHT$(STR$(Q),1);
150 NEXT I
160 PRINT: PRINT"▒▒▒▒ SOLL NOCH MEHR BERECHNET WERDEN? (J/N)"
170 GET Z$: IF Z$="" THEN GOTO 170
180 IF Z$="J" THEN GOTO 20
190 END

READY.
```

```
ZAEHLER? 1.7

NENNER? 3.9


ZAHL DER NACHKOMMASTELLEN? 25


1.7 / 3.9 = 0.4358974358568506339378797
```

Geburtstagswahrscheinlichkeit

Wenn Sie bei einer Party mit ca. 60 Gästen erklären, daß mindestens zwei von ihnen am selben Tag Geburtstag haben, werden die meisten dies vermutlich bezweifeln. Tatsächlich ist es aber nahezu sicher, daß diese Behauptung richtig ist, die Wahrscheinlichkeit für dieses Ereignis also sehr nahe bei 1 liegt!

Wie läßt sich so etwas berechnen? Gehen wir davon aus, daß die Geburten gleichmäßig über das Jahr verteilt sind. (Schaltjahre wollen wir auch unberücksichtigt lassen, da sie das Ergebnis ohnehin kaum beeinflussen würden).

Bei Wahrscheinlichkeitsberechnungen ist es häufig einfacher - bzw. die einzige Möglichkeit - , die Wahrscheinlichkeit für das komplementäre Ereignis zu berechnen und diesen Wert dann von 1 zu subtrahieren.

Anstatt die gesuchte Wahrscheinlichkeit direkt zu berechnen, ermitteln wir auch hier die Wahrscheinlichkeit, daß von N Personen alle an <u>verschiedenen</u> Tagen Geburtstag haben. Nennen wir diese Wahrscheinlichkeit Q, so ist die gesuchte Wahrscheinlichkeit dann P = 1-Q.

Bei zwei Menschen beträgt die Wahrscheinlichkeit, daß beide verschiedene Geburtstage haben: 364/365.

Nachdem zwei Daten belegt sind, ist die Wahrscheinlichkeit, daß eine dritte Person einen anderen Geburtstag hat, als die beiden ersten: 363/365.

Um die Wahrscheinlichkeit zu erhalten, daß alle drei Personen verschiedene Geburtstage haben, multipliziert man die einzelnen Wahrscheinlichkeiten (da diese ja als voneinander unabhängige Ereignisse betrachtet werden) und erhält:

Q = 365/365 * 364/365 * 363/365 = .9918 und somit
P = 1 - Q = .0082.

Allgemein beträgt die Wahrscheinlichkeit Q, daß von N Personen alle einen anderen Geburtstag haben:

$$\frac{365}{365} \quad \frac{364}{365} \quad \ldots \quad \frac{365-N+1}{365}$$

```
10 REM............................GEBURTSTAGS-WAHRSCHEINLICHKEIT
20 PRINT"▒▒ ▒ WAHRSCHEINLICHKEIT, DASS 2 PERSONEN "
30 PRINT" ▒                                      "
40 PRINT" ▒   AM GLEICHEN TAG GEBURTSTAG HABEN   "
50 REM
60 PRINT"▒▒▒▒▒▒▒▒▒▒▒▒ ";: INPUT"WIEVIELE PERSONEN";N$
70 IF N$=CHR$(13) THEN END
80 N=VAL(N$)
90 Q=1
100 FOR I=1 TO N
110 Q=Q*(365-I)/365
120 NEXT I
130 PRINT"▒▒ WAHRSCHEINLICHKEIT:                   ";
135 PRINT"▒ ";SPC(19);INT(1000*(1-Q)+.5)/1000
140 GOTO 60

READY.
```

```
                    WAHRSCHEINLICHKEIT, DASS 2 PERSONEN

                      AM GLEICHEN TAG GEBURTSTAG HABEN




                    WIEVIELE PERSONEN? 60

                    WAHRSCHEINLICHKEIT: .995
```

Z a h l e n u m r e c h n u n g e n

Umrechnung von Binärzahlen in Dezimalzahlen

Das binäre (oder duale) Zahlensystem (binär bzw. dual = 2) verwendet nur die beiden Ziffern 0 und 1 (vgl. Kapitel 9).

Genau wie im Dezimalsystem bestimmt auch im Binärsystem die Stelle innerhalb einer Zahl, an der eine Ziffer des Systems steht, ihren Wert. So läßt sich z.B. die Dezimalzahl 321 darstellen als:

1 * 1 + 2 * 10 + 3 * 100

Im Dezimalsystem werden die Ziffern Ihrem Stellenwert entsprechend mit Potenzen von 10 multipliziert: 10↑0 = 1, 10↑1 = 10, 10↑2 = 100 usw.

Im Binärsystem werden stattdessen Potenzen von 2 verwendet: 2↑0 = 1, 2↑1 = 2, 2↑3 = 8 usw.

Die Binärzahl 1101 wird demnach folgendermaßen (von rechts nach links) in eine Dezimalzahl umgerechnet:

```
1101 = 1 * 2↑0 + 0 * 2↑1 + 1 * 2↑2 + 1 * 2↑3
     = 1 * 1   + 0 * 2   + 1 * 4   + 1 * 8
     = 13
```

Ein Programm, das nach diesem Verfahren eine Binärzahl in eine Dezimalzahl umwandelt, muß also der Reihe nach die einzelnen Ziffern der Binärzahl mit den jeweils um eins erhöhten Potenzen zur Basis zwei multiplizieren. Dazu muß die Binärzahl aber zuerst einmal in einen String umgewandelt werden. Das folgende Programm arbeitet in der soeben beschriebenen Weise:

```
10 REM............................UMRECHNUNG BINAER - DEZIMAL (1)
20 PRINT"■■ ■ UMRECHNUNG: BINAER- IN DEZIMALZAHL "
30 PRINT"■ (TASTE 'E' -> PROGRAMMENDE)"
40 D=0: E=0: OK=0
50 PRINT"■■■■■■■";
60 INPUT" BINAERZAHL";B$
70 IF B$="E" THEN PRINT"■■■■": END
80 REM...............................................BERECHNUNG
90 FOR I=LEN(B$) TO 1 STEP -1
100 M$=MID$(B$,I,1)
110 IF (M$<"0" OR M$>"1") THEN OK=-1: GOTO 150
120 D=D+VAL(M$)*2↑E
130 E=E+1
140 NEXT I
150 IF OK=-1 THEN PRINT"■■■■ *** FALSCHE EINGABE!": PRINT: GOTO 40
160 REM..................................AUSGABE DES ERGEBNISSES
170 PRINT"■■ DEZIMALZAHL =";D;"■                    "
180 PRINT: GOTO 40

READY.
```

Ein anderes Verfahren zur Umrechnung von Binärzahlen in Dezimalzahlen wird durch die folgende Darstellung verdeutlicht und funktioniert so:

```
1         1          1         0         0        1

         + 1
1 * 2 =   2          + 1
         3 * 2 =    6          + 0
                    7 * 2 =   14          + 0
                             14 * 2 =  28          + 1
                                       28 * 2 =    56
                                                   57
                                                   ==
```

Die Umrechnung erfolgt schrittweise von links nach rechts.

1. Schritt: Die erste Ziffer der Binärzahl wird mit 2 multipliziert. Handelt es sich um eine einstellige Binärzahl, so ist die Berechnung beendet: das Ergebnis lautet entweder 0 oder 2.

   Ansonsten geht es weiter mit dem

2. Schritt: Zu dem Ergebnis wird die nächste Ziffer der Binärzahl hinzugezählt. Jetzt wird dieses Ergebnis mit 2 multipliziert.

   Sind alle Stellen abgearbeitet, so ist die Umrechnung beendet. Wenn nicht, wird der zweite Schritt wiederholt.

Dieser Algorithmus liegt dem nächsten Programm zugrunde. Zusätzlich werden auch die Zwischenergebnisse ausgegeben.

```
10 REM ...........................UMRECHNUNG BINAER - DEZIMAL (2)
20 PRINT"   UMRECHNUNG: BINAER- IN DEZIMALZAHL "
30 PRINT"  ('E' -> PROGRAMMENDE)"
40 D=0: F=-1: K=0
50 PRINT"        ";
60 INPUT" BINAERZAHL";B$
70 IF B$="E" THEN PRINT"    ": END
80 REM.......................................BERECHNUNG
90 FOR I=1 TO LEN(B$)
100 M$=MID$(B$,I,1)
110 IF (M$<"0"OR M$>"1") THEN F=1: K=K+1: GOTO 130
120 D=2*D+VAL(M$)
130 NEXT I
140 PRINT
145 REM.......................................EINGABE ZULAESSIG?
150 IF F<=0 THEN PRINT"                                          "
160 IF F>0 THEN PRINT" FALSCHE EINGABE! (";
170 IF F>0 THEN PRINT K;" ZIFFERN FALSCH!)";: GOTO 40
175 REM..............................................ERGEBNIS
180 PRINT" DEZIMALZAHL =";D;"                 "
190 GOTO 40

READY.
```

Umrechnung von Dezimalzahlen in Binärzahlen

Das folgende Verfahren stellt die Umkehrung des zuvor angegebenen Algorithmus zur stellenweisen Umrechnung von Binärzahlen dar:

Durch fortgesetzte - ganzzahlige - Division einer Dezimalzahl durch zwei erhält man unter entsprechender Berücksichtigung des Divisionsrestes die Binärzahl. Die Berechnung ist beendet, wenn die Division den Wert Null ergibt. Die Binärzahl wird hierbei von rechts nach links aufgebaut.

Mit dem folgenden Beispiel wird das Vorgehen erläutert:

Die Zahl 57 soll als Binärzahl dargestellt werden. Die Division durch 2 ergibt 28, Rest 1. Dies ist die erste - rechte - Ziffer der Binärzahl.

Die Division von 28 durch 2 ergibt 14, Rest 0. Dies ist die zweite Ziffer der Binärzahl, die somit bisher 01 heißt.

Entsprechend geht man vor, bis sich im letzten Schritt unseres Beispiels bei der Division von 1 durch 2 der Wert 0 ergibt (mit dem Rest 1), womit die Berechnung abgeschlossen ist.

Die gesamte Berechnung ist im folgenden in übersichtlicher Form dargestellt. Das nachstehende Programm rechnet Dezimalzahlen nach dem hier beschriebenen Verfahren um.

```
57 : 2 = 28   Rest  1

28 : 2 = 14   Rest  0

14 : 2 =  7   Rest  0

 7 : 2 =  3   Rest  1

 3 : 2 =  1   Rest  1

 1 : 2 =  0   Rest  1   1   1   0   0   1
```

```
10 REM................................UMRECHNUNG DEZIMAL - BINAER
15 PRINT" UMRECHNUNG VON DEZIMAL-IN BINAERZAHLEN"
20 B$="": R$=""
30 INPUT" DEZIMALZAHL";D$
40 IF D$="E" THEN PRINT"": END
50 D=VAL(D$)
60 Q=INT(D/2)
70 R$=R$+STR$((D/2-Q)*2)
80 D=Q
90 IF Q>0 THEN GOTO 60
100 FOR I=LEN(R$) TO 1 STEP -1
110 B$=B$+MID$(R$,I,1)
120 NEXT I
130 PRINT" BINAER --> ";B$;"          "
140 PRINT: GOTO 20
150 IF D>0 THEN GOTO 50
160 GOTO 60

READY.
```

Umwandlung von Dezimalzahlen in Zahlen mit beliebiger Basis und umgekehrt

Das nächste Programm ist nicht auf die Umrechnung von Binärzahlen beschränkt, sondern kann eine Dezimalzahl in eine Zahl mit einer beliebigen Basis zwischen zwei und sechszehn umrechnen, und umgekehrt. Eine größere Basis ist nicht sinnvoll, weil das Hexadezimalsystem mit der Basis 16 die größte Basis verwendet, die eine praktische Bedeutung hat.

Die Umwandlung einer Dezimalzahl in eine Zahl zur Basis G geschieht nach folgendem Verfahren:

Aus der Darstellung

$$N = A \quad G \; + A \quad\quad G \quad\quad + \ldots A \quad G \; + A \quad G$$

erhält man die G-adischen Ziffern (also die Ziffern zur Basis G) durch fortgesetzte Division durch die Basis G. Der Divisionsrest ergibt - mit A beginnend - die G-adischen Ziffern. Die Umrechnung ist beendet, wenn der ganzzahlige Teil des Quotienten N/G gleich Null ist.

Zur Umwandlung einer Zahl mit der Basis G in eine Dezimalzahl wird die Horner-Darstellung benutzt. Die Auswertung der Klammerausdrücke erfolgt dabei von innen nach außen:

$$N = (\ldots (a \quad G + a \quad\quad)G + a \quad\quad)G + \ldots a \;)G + a$$

```
10 REM........................................ZAHLENUMWANDLUNGEN
15 PRINT"{CLR}{DOWN} {RVS}          ZAHLENUMWANDLUNGEN          "
20 PRINT"{DOWN}{DOWN} MIT DIESEM PROGRAMM KOENNEN SIE"
30 PRINT" DEZIMALZAHLEN IN ZAHLEN MIT BELIEBIGER"
35 PRINT" BASIS (>1 UND <=16) UMWANDELN LASSEN."
40 PRINT"{DOWN} BITTE WAEHLEN SIE:"
45 CZ=12: CS=5: GOSUB 1000
50 PRINT"1  DEZIMAL --> BELIEBIGE BASIS"
55 CZ=13: CS=5: GOSUB 1000
60 PRINT"2  BELIEBIGE BASIS --> DEZIMAL"
65 CZ=14: CS=5: GOSUB 1000
70 PRINT"0  ENDE"
80 GET A$
85 IF A$="" THEN GOTO 80
90 IF A$="0" THEN PRINT"{DOWN}{DOWN}{DOWN}": END
95 IF (A$<>"1" AND A$<>"2") THEN GOTO 80
100 ON VAL(A$) GOTO 110,290
110 REM...............................DEZIMAL --> BELIEBIGE BASIS
120 PRINT"{CLR}{DOWN} {RVS} DEZIMALZAHL -> BELIEBIGE BASIS <= 16 "
130 Y$="0123456789ABCDEF"
140 INPUT"{HOME}{DOWN}{DOWN}{DOWN} BASIS:        "; G
150 IF G=0 THEN PRINT"{DOWN}{DOWN}{DOWN}{DOWN}{DOWN}{DOWN}{DOWN}{DOWN}{DOWN}": END
160 INPUT"{DOWN} DEZIMALZAHL:"; N
190 Z$=""
200 PRINT"{DOWN}{DOWN}          --> ";
210 FOR I=0 TO 255
220 R=N-G*INT(N/G)
230 Z$=MID$(Y$,R+1,1)+Z$
240 N=INT(N/G)
250 IF N=0 THEN GOTO 270
260 NEXT I
270 PRINT Z$;"          "
280 GOTO 130
```

```
290 REM..............................BELIEBIGE BASIS --> DEZIMAL
300 PRINT"▒▒ ▒ ZAHL MIT BASIS <= 16 --> DEZIMALZAHL "
310 Y$="0123456789ABCDEF"
320 INPUT"▒▒▒▒ BASIS:        ";G
325 IF G=0 THEN PRINT"▒▒▒▒▒▒▒▒▒": END
330 INPUT"▒ ZAHL:          ";Z$
340 PRINT"▒▒ DEZIMAL -->";
350 N=0
360 FOR I=1 TO LEN(Z$)
400 T$=MID$(Z$,I,1)
410 FOR K=1 TO LEN(Y$)
420 IF T$<>MID$(Y$,K,1) THEN 450
430 N=N*G+K-1
440 GOTO 460
450 NEXT K
460 NEXT I
470 PRINT N;"▒▒                          "
480 GOTO 320
490 END
1000 CZ=1024+40*CZ
1010 POKE 209,CZAND255
1020 POKE 210,CZ/256
1030 POKE 211,CS
1040 RETURN

READY.
```

```
 DEZIMALZAHL -> BELIEBIGE BASIS <= 16

BASIS:      ? 13

DEZIMALZAHL:? 365962


       --> CA75C


  ZAHL MIT BASIS <= 16 --> DEZIMALZAHL

 BASIS:       ? 16

 ZAHL:        ? 9ABCDEF


 DEZIMAL --> 162254319



READY.
```

L o g i k - F o r m e l

Das vorliegende Programm wertet eine logische Formel mit drei Variablen - A, B und C - aus. Die möglichen Kombinationen der Wahrheitswerte sind in den DATA-Zeilen angegeben. Das Programm ermittelt, für welche dieser Kombinationen die Verknüpfung der Variablen insgesamt gilt, also den Wahrheitswert -1 hat.

Die Variablen können als Stellvertreter für beliebige Aussagen betrachtet werden. Analysieren wir z.B. den Satz:

"Kunden erhalten 3% Rabatt, wenn sie bereits länger als 5 Jahre zur Kundschaft gehören, oder wenn sie Waren im Wert von mehr als DM 1000.- bestellt haben und nicht mit Zahlungen im Rückstand sind."

Die Frage ist, in welchen Fällen ein Kunde den Rabatt erhält. Dazu werden drei Aussagen gemacht:

A Jemand ist länger als 5 Jahre Kunde.
B Jemand hat Waren im Wert von mahr als DM 1000.- bestellt.
C Jemand ist mit Zahlungen nicht im Rückstand.

Übersetzt in einen logischen Ausdruck mit den angegebenen Variablen lautet der obige Satz:

A ODER B UND C

Zu beachten ist, daß das logische UND stärker bindet als das ODER, d.h. die Aussagenverknüpfung wird so ausgewertet, als stünden die durch UND verknüpften Variablen in Klammern:

A ODER (B UND C)

Diese Aussagenverknüpfung gilt also z.B. dann, wenn alle Teilaussagen wahr sind, d.h. jemand, der schon länger als 5 Jahre Kunde ist und Waren im Wert von mehr als DM 1000.- bestellt hat und mit Zahlungen nicht im Rückstand ist erhält selbstverständlich den Rabatt. Dies gilt jedoch auch für jemand, der noch nicht so lange zur Kundschaft gehört, aber auch Waren im Wert von mehr als DM 1000.- bestellt und nicht mit Zahlungen im Rückstand ist usw.

```
10 REM.............................................LOGIK-FORMEL
20 PRINT"▯▣ ▣                    LOGIK-FORMEL              "
30 REM.....................DEFINITION EINER AUSSAGENVERKNUEPFUNG
40 DEF FN L(X)=A(X) OR (B(X) AND C(X))
50 REM.....ZAHL DER MOEGL. KOMBINATIONEN = 2↑(ZAHL DER VARIABLEN)
60 N=8
70 DIM A(N),B(N),C(N)
80 REM............................BERECHNUNG ALLER KOMBINATIONEN
90 PRINT: PRINT TAB(5);" A  B  C     A OR (B AND C)": PRINT
100 FOR I=1 TO N
110 READ A(I),B(I),C(I)
120 PRINT TAB(5);A(I);B(I);C(I);TAB(21);FN L(I);
130 IF FN L(I)=1 THEN PRINT" WAHR"
140 IF FN L(I)=0 THEN PRINT" FALSCH"
150 NEXT I
160 END
170 REM................................DATEN DER KOMBINATIONEN
180 DATA 1,1,1
190 DATA 1,1,0
200 DATA 1,0,1
210 DATA 1,0,0
220 DATA 0,1,1
230 DATA 0,1,0
240 DATA 0,0,1
250 DATA 0,0,0

READY.
```

L i s s a j o u s - F i g u r e n

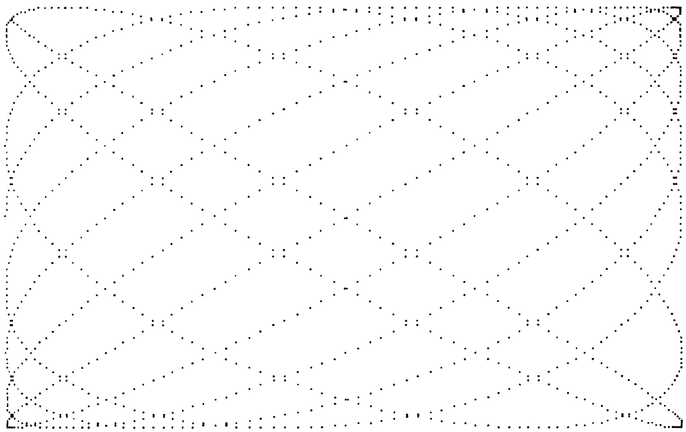
Das folgende Programm zeichnet Lissajous-Figuren auf den Bildschirm.

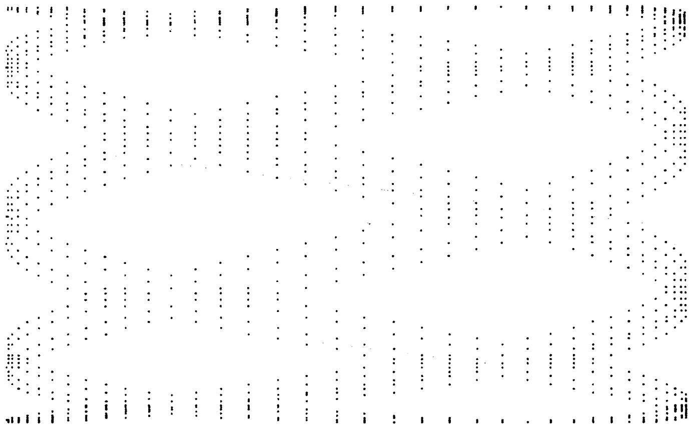
Lissajous (1822 - 1880) war ein französischer Physiker, der sich mit visuellen Darstellungen von Schwingungen beschäftigte.

Bei der Überlagerung von zwei zueinander senkrecht stehenden Schwingungen entstehen Figuren, wie sie auf dem Bildschirm dargestellt werden.

```
10 REM.........................................LISSAJOUS-FIGUREN
20 PRINT"□▣ ▣           LISSAJOUS-FIGUREN          "
30 PRINT: INPUT" NUMMER ZWISCHEN 10 UND .....";E
35 REM.........................................GRAFIK EINSCHALTEN
40 POKE 53248+17,59: POKE 53248+24,24
45 POKE 53280,1
50 FOR I=1024 TO 2023: POKE I,1: NEXT I
55 FOR I=8192 TO 16191: POKE I,0: NEXT I
60 FOR I=0 TO 3000
65 Y=99+99*SIN(I/99*π)
70 X=159+159*COS(I/E*π)
75 IF X<=0 THEN X=.01
80 IF Y<=0 THEN Y=.01
85 IF X>=319 THEN X=319
90 IF Y>=199 THEN Y=199
100 GOSUB 1000
105 GET A$: IF A$<>"" THEN GOTO 130
110 NEXT I
120 GET A$: IF A$="" THEN GOTO 120
130 POKE 53248+17,155: POKE 53248+24,21
140 POKE 53280,14: PRINT"□"
150 END
999 REM.............................................PUNKT SETZEN
1000 XK=8*INT(X/8)
1010 YK=320*INT(Y/8)+INT((Y/8-INT(Y/8))*8)
1020 S=8192+XK+YK
1030 EX=2↑(7-INT((X/8-INT(X/8))*8))
1040 POKE S,PEEK(S) OR EX
1050 RETURN

READY.
```





Die Maus im Labyrinth

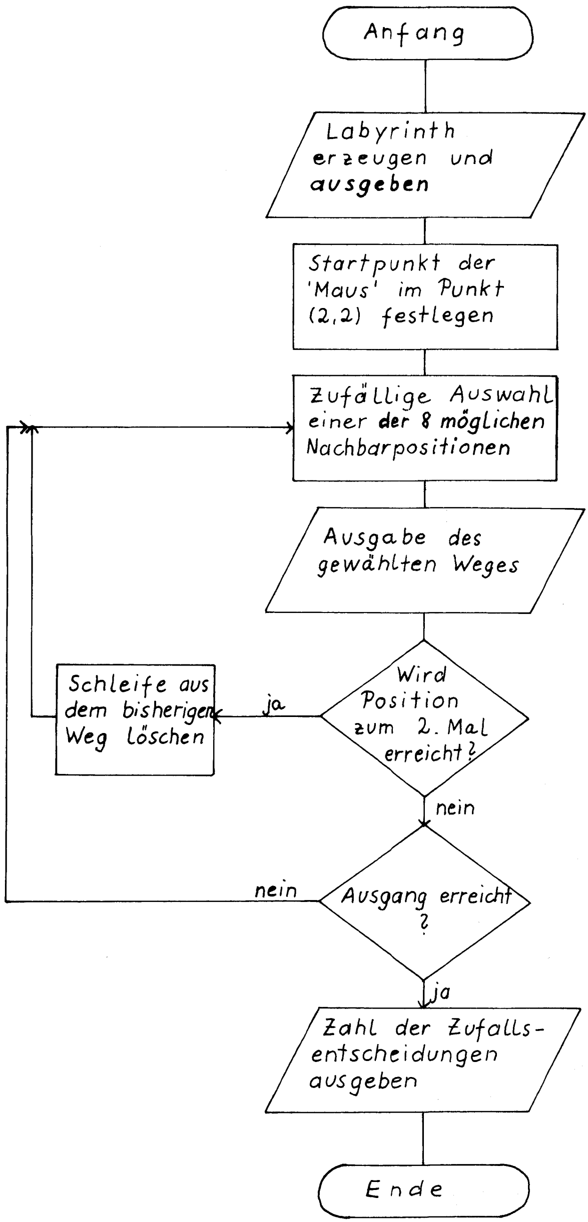
Dieses Programm simuliert den Versuch einer "Maus", aus einem Labyrinth herauszukommen, das nur einen einzigen Ausgang besitzt.

Das Labyrinth ist durch die Daten in den DATA-Zeilen fest vorgegeben. Diese Daten werden mit READ in das zweidimensionale Feld A eingelesen. Zu Beginn befindet sich die "Maus" im Punkt (2,2). Von den bis zu acht benachbarten Positionen wird (mit Hilfe der RND-Funktion) eine zufällig ausgewählt. Wenn die Maus einen Umweg macht, also einen Punkt zum zweiten Mal erreicht, wird dieser Umweg gelöscht. Die Informationen über den Weg der Maus werden im Feld A notiert.

Erreicht die Maus den Ausgang (Punkt 13,10), so wird die Zahl der benötigten Versuche, d.h. Zufallsentscheidungen, ausgegeben.

Das Programm läuft zwar völlig selbständig ab, bietet aber dennoch Möglichkeiten zum Experimentieren: man kann z.B. leicht das vorgegebene Labyrinth verändern, eventuell das Labyrinth auch vom Programm selbst erstellen lassen.

Interessanter noch dürfte es sein, die Strategie der Maus zu verändern, z.B. so, daß sie bei mehreren möglichen Nachbarpositionen die Richtung nicht völlig willkürlich ändert, sondern ihren Weg möglichst geradeaus fortsetzt.



Fluß-Diagramm: Maus im Labyrinth

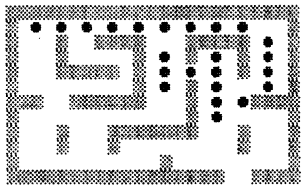
```
10 REM.........................................MAUS IM LABYRINTH
15 POKE 53280,0: POKE 53281,11
20 PRINT"[illegible]          MAUS IM LABYRINTH          "
30 DIM A(13,13)
40 FOR I=1 TO 12
50 READ A$
60 FOR J=1 TO 12
70 A(I,J)=-VAL(MID$(A$,J,1))
80 NEXT J
90 NEXT I
100 C=0
110 PRINT"[illegible]";
120 X=2: Y=2: A(2,2)=1
130 Z=0
140 N=INT(8*RND(1))+1
150 C=C+1
159 REM.........................................WAHL DES WEGES
160 ON N GOTO 170,180,190,200,210,220,230,240
170 I=X-1: K=Y-1: GOTO 250
180 I=X-1: K=Y: GOTO 250
190 I=X-1: K=Y+1: GOTO 250
200 I=X: K=Y+1: GOTO 250
210 I=X+1: K=Y+1: GOTO 250
220 I=X+1: K=Y: GOTO 250
230 I=X+1: K=Y-1: GOTO 250
240 I=X: K=Y-1
250 IF A(I,K)<0 THEN GOTO 140
260 IF Z THEN GOTO 590
270 GOSUB 350
280 IF A(I,K)>0 THEN GOTO 530
290 A(I,K)=N
300 X=I: Y=K
310 IF X>12 THEN GOTO 630
320 GOTO 160
330 GOSUB 350
```

```
340 GOTO 630
350 PRINT"▚▚▚▚▚▚▚";
360 FOR L=1 TO 12
370 FOR M=1 TO 12
380 J=A(L,M)+3
390 IF J<4 THEN GOTO 410
400 J=4
410 ON J GOTO 420,440,460,480
415 REM..................................................LABYRINTH
420 PRINT" ▚ ■";
430 GOTO 490
440 PRINT"▚  ■";
450 GOTO 490
460 PRINT"  ";
470 GOTO 490
480 PRINT"◣ ●╥";
490 NEXT M
500 PRINT
510 NEXT L
520 RETURN
530 N=A(X,Y): A(X,Y)=0
540 A(X,Y)=0
550 Z=-1
560 P=I: Q=K
570 ON N GOTO 210,220,230,240,170,180,190,200
580 STOP
590 X=I: Y=K
600 IF (P=I AND Q=K) THEN GOTO 130
610 N=A(I,K): A(I,K)=0
620 GOTO 570
630 PRINT"▚";: FOR R=1 TO 19: PRINT"▚";: NEXT R
635 PRINT" ◣MAUS AM AUSGANG!╥"
640 PRINT: PRINT" NACH";C;"ZUFALLSENTSCHEIDUNGEN."
650 END
```

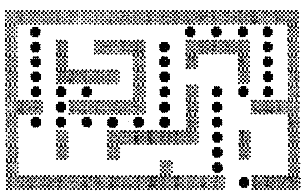
```
659 REM....................................DATEN FUER LABYRINTH
660 DATA 211111111111
670 DATA 200000000002
680 DATA 202011021102
690 DATA 202002020202
700 DATA 202112000202
710 DATA 200002020002
720 DATA 210111020011
730 DATA 200000020002
740 DATA 202021110202
750 DATA 202020000202
760 DATA 200000200002
770 DATA 211111111011

READY.
```

MAUS IM LABYRINTH



MAUS IM LABYRINTH



MAUS AM AUSGANG!
NACH 128 ZUFALLSENTSCHEIDUNGEN.
READY.

Monte Carlo-Berechnung von Pi

Mit Hilfe von Simulationen lassen sich viele Probleme lösen, für die es eine exakte Lösung nicht gibt, oder für die eine solche Berechnung viel zu aufwendig wäre - selbst für einen Computer. Ein Beispiel aus dem mathematischen Bereich ist die Berechnung mehrfacher Integrale. Die Bezeichnung 'Monte Carlo' soll daran erinnern, daß der Zufall bei diesen Berechnungen seine Hand im Spiel hat.

Das Simulationsprogramm zur Berechnung der Kreiszahl basiert auf folgender Überlegung:

In ein Quadrat mit der Seitenlänge r=1 wird ein Kreis gezeichnet. Die Fläche des Kreises ist $(\pi \cdot r\uparrow 2)$ oder, da r=1 ist, $\pi \cdot 4$, während die Fläche des Quadrats $(2 \cdot r)\uparrow 2 = 4$ ist.

Betrachten wir einen zufällig gewählten Punkt innerhalb des Quadrats, so ist die Wahrscheinlichkeit p, daß er sich auch im Kreis befindet, offensichtlich proportional zu dem Verhältnis der Flächen von Kreis und Quadrat, d.h.:

$$p = \frac{(\pi \cdot r\uparrow 2)}{(2 \cdot r)\uparrow 2} = \frac{\pi}{4} \qquad \text{also: } \pi = 4 \cdot p$$

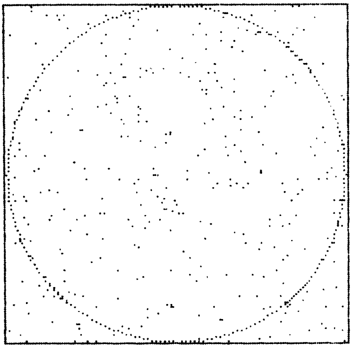
Wählt man nun aus dem Quadrat willkürlich Punkte aus und bildet das Verhältnis der im Kreis befindlichen zur Gesamtzahl der Punkte, erhält man einen Näherungswert für π . Dieser wird umso genauer, je größer die Anzahl der Punkte ist.

```
10 REM.............................................PI-BERECHNUNG
20 PRINT"□▣ ▣     MONTE CARLO - BERECHNUNG VON PI    ": PRINT
30 PRINT" ES WIRD HIERBEI DAS VERHAELTNIS DER"
40 PRINT" INNERHALB BZW. AUSSERHALB DES KREISES"
50 PRINT" LIEGENDEN PUNKTE BERECHNET."
60 PRINT"▣ NACH ZEICHNEN EINES KREISES UND EINES   QUADRATS";
65 PRINT" HAT MAN MIT 'U' DIE MOEGLICH-  KEIT, DIE AKTUELLE";
70 PRINT" NAEHERUNG ABZU-      FRAGEN."
75 PRINT:INPUT" WEITER (J/N)";A$
80 IF A$="N" THEN END
90 REM........................................GRAFIK EINSCHALTEN
95 V=53248
100 POKE V+17,59: POKE V+24,24
110 FOR I=1024 TO 2023: POKE I,1: NEXT I
120 FOR I=8192TO 16191: POKE I,0: NEXT I
125 REM...........................................ZEICHNEN KREIS
130 FOR I=-60 TO 60 STEP 2
140 X=160+I
150 Y1=100-SQR(6241-I*I)
160 Y2=100+SQR(6241-I*I)
170 Y=100+I
180 X1=160-SQR(6241-I*I)
190 X2=160+SQR(6241-I*I)
200 XX=X: YY=Y1: GOSUB 60000
205 XX=X: YY=Y2: GOSUB 60000
210 XX=X1: YY=Y: GOSUB 60000
215 XX=X2: YY=Y: GOSUB 60000
220 NEXT I
```

```
225 REM.........................................ZEICHNEN QUADRAT
230 FOR XX=80 TO 240
235 YY=20: GOSUB 60000
240 NEXT XX
245 FOR YY=20 TO 179
250 XX=240: GOSUB 60000
255 NEXT YY
260 FOR XX=240 TO 80 STEP-1
265 YY=179: GOSUB 60000
270 NEXT XX
275 FOR YY=179 TO 20 STEP-1
280 XX=79: GOSUB 60000
285 NEXT YY
290 KX=RND(1)*2-1: KY=RND(1)*2-1
295 IF (KX*KX+KY*KY)<1 THEN IN=IN+1
300 N=N+1
305 XX=160+KX*79: YY=100+KY*79: GOSUB 60000
310 REM ......................................BERECHNUNG VON PI
315 PI=INT(4*IN/N*1000000+.5)/1000000
320 DI=(PI/π-1)*100
330 PR=INT(DI*1000+.5)/1000
335 REM ......................................AUSGABE DER WERTE
340 GETA$: IF A$<>"U" THEN GOTO 390
345 POKE V+17,155: POKE V+24,21
350 PRINT"{CLR}{DOWN}{DOWN} AKTUELLE NAEHERUNG FUER PI = ";PI
355 PRINT"{DOWN} ABWEICHUNG VOM WAHREN WERT = ";PR;"%"
360 PRINT"{DOWN} ANZAHL DER BERECHNUNGEN     = ";N
365 PRINT"{DOWN}{DOWN}{DOWN}{DOWN} BERECHNUNG FORTSETZEN (J/N)"
370 GET C$: IF C$="" OR (C$<>"J" AND C$<>"N") THEN GOTO 370
375 IF C$="N" THEN GOTO 400
380 POKE V+17,59: POKE V+24,24:FOR I=1024 TO 2023:POKE I,1:NEXT I
390 GOTO 290
400 END
```

```
59999 REM...........................UNTERPROGRAMM PUNKT ZEICHNEN
60000 A=INT(XX/8)*8+320*INT(YY/8)+INT((YY/8-INT(YY/8))*8)
60010 B=2↑(7-INT((XX/8-INT(XX/8))*8))
60020 C=8192+A
60030 POKE C,PEEK(C) OR B
60040 RETURN

READY.
```



```
AKTUELLE NAEHERUNG FUER PI =  3.142091

ABWEICHUNG VOM WAHREN WERT =  .016 %

ANZAHL DER BERECHNUNGEN    =  373



BERECHNUNG FORTSETZEN (J/N)

READY.
```

L I F E

LIFE (= Leben) ist ein Simulationsspiel, das von John Conway erfunden wurde. Als Spielfeld benötigt man lediglich ein Blatt kariertes Papier und einen Bleistift. In den Kästchen werden 'Lebewesen' in beliebiger Anordnung dargestellt. Diese stellen die ursprüngliche 'Bevölkerung' dar, die sich in jeder 'Generation' dadurch verändern kann, daß Lebewesen sterben oder neue Lebewesen entstehen.

Die Regeln für dieses 'Lebensspiel' sind sehr einfach:

1. Jedes Element mit 2 oder 3 Nachbarn überlebt in der betreffenden Generation.

2. Jedes Element mit mehr als 3 Nachbarn stirbt.

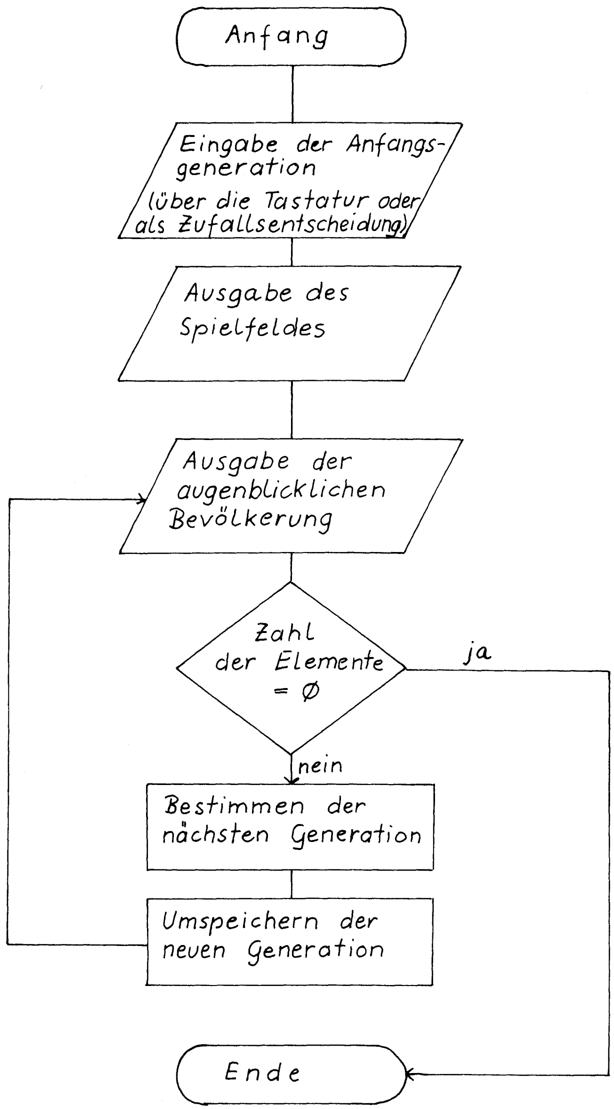
3. In jedem Kästchen, das genau 3 Nachbarn hat, entsteht ein neues Element.

Das Spiel bietet sich direkt dazu an, daß man es mit dem Computer spielt. Das Programm ist so ausgelegt, das man die Ausgangspopulation sowohl selbst eingeben kann, oder aber den Computer eine zufällige Anfangsgeneration erzeugen läßt. Danach läuft das Lebensspiel nach den genannten Regeln von ganz allein ab.

Insbesonders bei einer zufälligen Ausgangspopulation ist es sehr reizvoll, festzustellen, wie sich auf Grund der Regeln schon sehr bald oft sehr interessante regelmäßige Muster bilden.

Im Anschluß an das Programm-Listing ist eine Ausgangspopulation abgebildet, die als Muster für eigene Versuche dienen kann.

Von einer anderen Ausgangsfigur finden Sie die Abbildungen sämtlicher sich daraus ergebenden Generationen.



Fluß-Diagramm: Life-Spiel

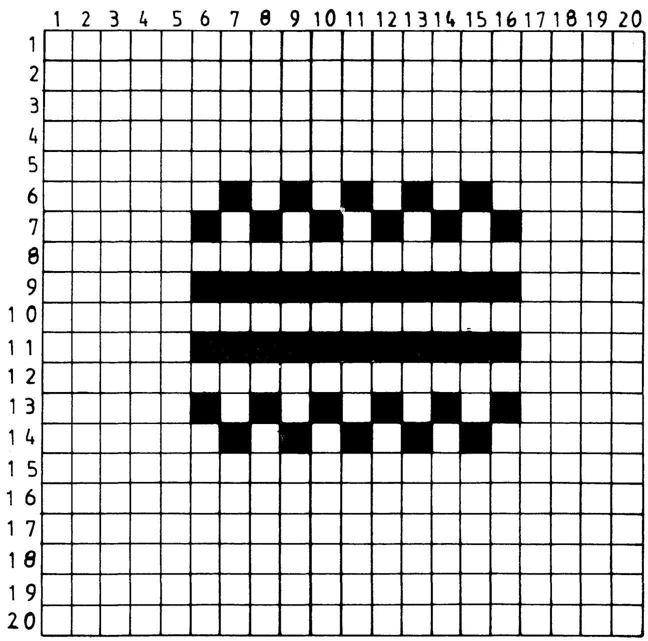
```
5 REM.............................................GAME OF LIFE
10 PRINT"⌐◙ ◙               GAME OF LIFE          "
15 PRINT"◙ 1. EIN LEBEWESEN STIRBT, WENN WENIGER"
20 PRINT"    ALS ZWEI ODER MEHR ALS DREI LEBE- "
25 PRINT"    WESEN IN UNMITTELBARER NACHBARSCHAFT"
30 PRINT"⌐    LEBEN."
40 PRINT:PRINT"◙ 2. EIN NEUES LEBEWESEN ENTSTEHT AUF"
45 PRINT"    EINEM FELD, WENN AUF ALLEN NACHBAR-"
50 PRINT"    FELDERN INSGESAMT DREI LEBEWESEN "
55 PRINT"    EXISTIEREN."
60 PRINT"◙◙":INPUT" WIEVIELE LEBEWESEN";N
65 DIM A(22,22), A1(22,22), B(22,22)
70 REM ....................................SETZEN DER LEBEWESEN
80 PRINT: INPUT" 1. GENERATION SELBST EINGEBEN (J/N)";A$
90 IF A$="J" THEN GOTO 180
100 REM...............ZUFAELLIGE VERTEILUNG DER ANFANGSGENERATION
110 FOR I=1 TO N
120 Z=INT(RND(1)*20+1)
130 S=INT(RND(1)*20+1)
140 A(Z,S)=1
150 A1(Z,S)=1
160 NEXT I
170 GOTO 260
180 REM..............................EINGABE UEBER DIE TASTATUR
190 FOR I=1 TO N
200 PRINT I;". ZEILE, SPALTE";
210 INPUT Z,S
220 IF Z>20 OR S>20 THEN PRINT"*** MAXIMUM = 20!": GOTO 200
230 A(Z,S)=1
250 NEXT I
260 REM.....................................SPIELFELD AUSGEBEN
270 PRINT"⌐◙◙◙◙";: POKE 53280,6: POKE 53281,6
280 FOR I=1 TO 20
290 PRINT"■◙                    ": REM 20 LEERZEICHEN
300 NEXT I
```

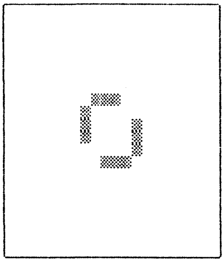
```
310 REM...............................AUSGABE DER GENERATIONEN
330 G=G+1
340 NN=0
350 PRINT"▒▒▒▒▒▒";TAB(22);"▒";G;"▮. GENERATION "
360 PRINT"▒▒▒▒";
370 FOR I=1 TO 20
380 FOR J=1 TO 20
390 IF A(I,J)<>1 THEN 450
400 PRINT"▒▒ ";
410 IF I>IMAX THEN IMAX=I
420 IF J>JMAX THEN JMAX=J
430 NN=NN+1
440 GOTO 460
450 PRINT"■▒ ";
460 NEXT J
470 PRINT
480 NEXT I
490 PRINT"▒▒▒▒▒▒▒▒▒▒▒";TAB(22);CHR$(154);
495 PRINT"▒ N =";NN;"▮    "
500 IF NN=0 THEN GOTO 890
510 REM........................BESTIMMEN DER NAECHSTEN GENERATION
520 FOR I=1 TO IMAX+1
530 FOR J=1 TO JMAX+1
540 IF A(I,J)=1 THEN GOTO 650
550 REM......................................PRUEFEN OB GEBURT
560 S=0
570 FOR L=I-1 TO I+1
580 FOR K=J-1 TO J+1
590 S=S+A(L,K)
600 NEXT K
610 NEXT L
620 IF S<>3 THEN GOTO 760
630 B(I,J)=1
640 GOTO 770
```

```
650 REM.......................................PRUEFEN OB TOD
660 S=-1
670 FOR L=I-1 TO I+1
680 FOR K=J-1 TO J+1
690 S=S+A(L,K)
700 NEXT K
710 NEXT L
720 IF S<2 THEN 760
730 IF S>3 THEN 760
740 B(I,J)=1
750 GOTO 770
760 B(I,J)=0
770 NEXT J
780 NEXT I
790 REM........................UMSPEICHERN DER NEUEN GENERATION
800 GL=0
810 FOR I=1 TO IMAX+1
820 FOR J=1 TO JMAX+1
840 A(I,J)=B(I,J)
850 NEXT J
860 NEXT I
880 GOTO 310
890 END

READY.
```

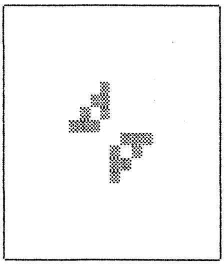
RASTER FÜR LIFE-SPIEL





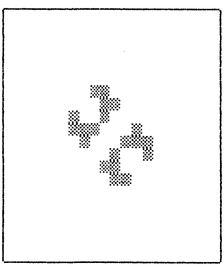
1. GENERATION

N = 12



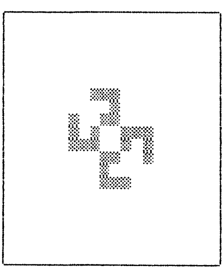
2. GENERATION

N = 16



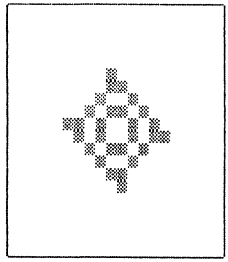
3. GENERATION

N = 20



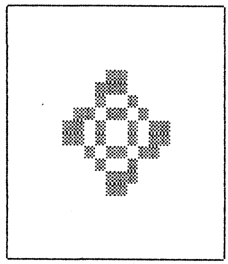
4. GENERATION

N = 24



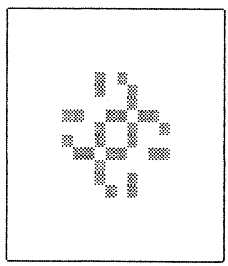
5. GENERATION

N = 28



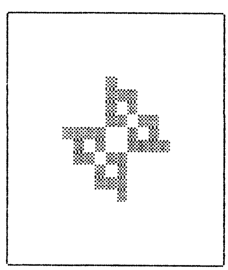
6. GENERATION

N = 36



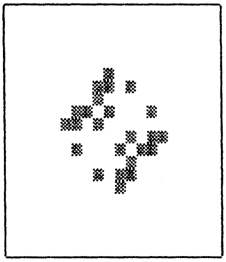
7. GENERATION

N = 28



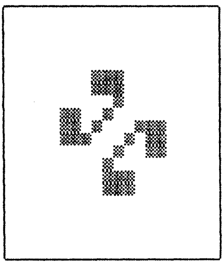
8. GENERATION

N = 32



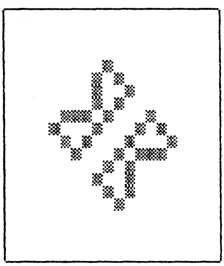
9. GENERATION

N = 24



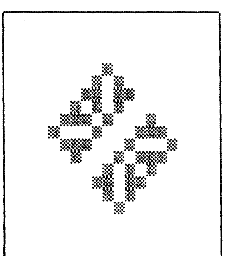
10. GENERATION

N = 32



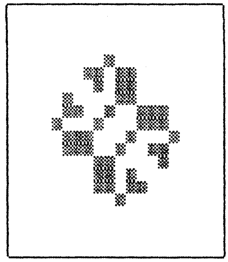
11. GENERATION

N = 32



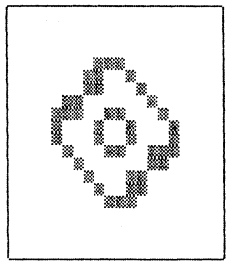
12. GENERATION

N = 40



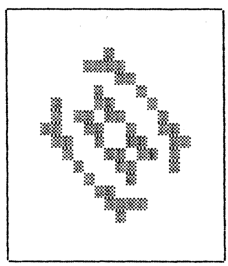
13. GENERATION

N = 44



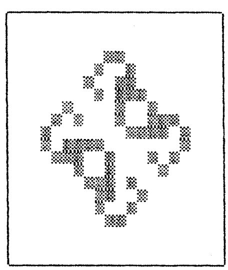
14. GENERATION

N = 44



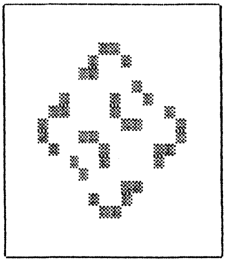
15. GENERATION

N = 52



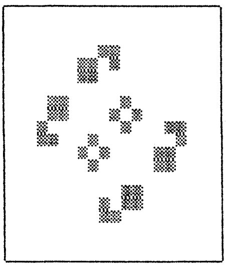
16. GENERATION

N = 52



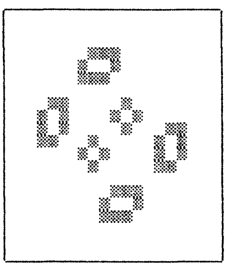
17. GENERATION

N = 36



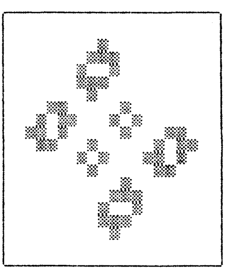
18. GENERATION

N = 36



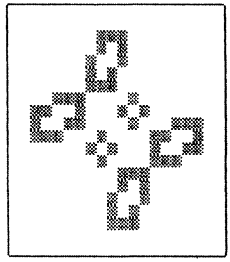
19. GENERATION

N = 40



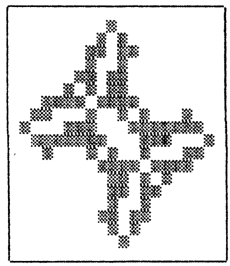
20. GENERATION

N = 48



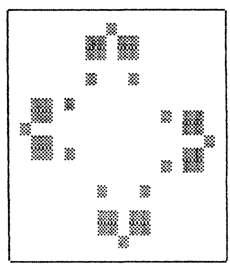
21. GENERATION

N = 56



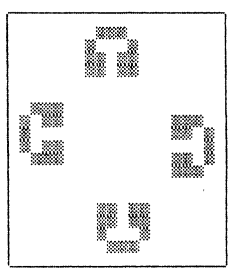
22. GENERATION

N = 84



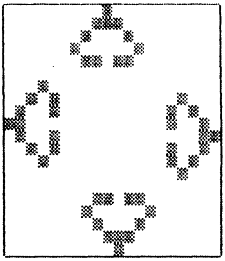
23. GENERATION

N = 44



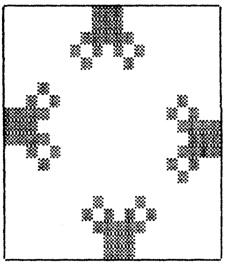
24. GENERATION

N = 52



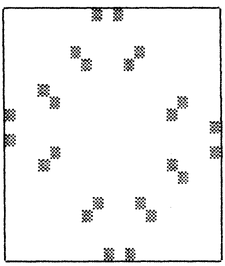
25. GENERATION

N = 48



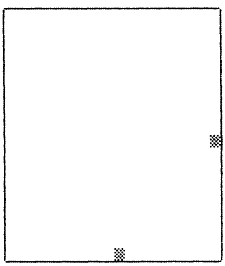
26. GENERATION

N = 68



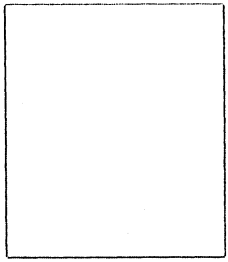
27. GENERATION

N = 24



28. GENERATION

N = 2



29. GENERATION

N = 0

Zinseszins - Berechnungen

Die Formel zur Berechnung des Endwertes eines Kapitals einschließlich Zinseszinsen lautet:

KN = KO * (1 + P/100)↑N

Dabei gelten folgende Bezeichnungen:

- KN Endkapital,
- KO Anfangskapital,
- P Zinssatz in % pro Zinsperiode,
- N Anzahl der Zinsperioden.

Löst man diese Gleichung nach den anderen Variablen auf, so kann man - wenn drei dieser Werte gegeben sind - den jeweils fehlenden berechnen:

$$KO = \frac{KN}{(1 + P/100)\uparrow N}$$

$$N = \frac{LOG(KN/KO)}{LOG(1 + P/100)}$$

P = ((KN/KO)↑(1/N) - 1) * 100

Weitere Programme zu wirtschaftlichen Themen (z.B. Optimierung, Lohnsteuerberechnung, Zeitreihenanalyse) sind in dem Buch: Elsing/Herrmann: "Wirtschaft auf dem Commodore 64" enthalten.

```
10 REM..................................ZINSESZINS-BERECHNUNGEN
20 PRINT"{CLR}{RVS} {HOME}       ZINSESZINS-BERECHNUNGEN        "
25 REM.........................................RUNDUNGSFUNKTION
30 DEF FN R(X) = INT(X*100+.5)/100
35 PRINT "{HOME}{DOWN 15} FUER DEN GESUCHTEN WERT 'RETURN'"
40 PRINT " EINGEBEN!"
45 PRINT "{HOME}{DOWN 5}";
50 INPUT " ENDKAPITAL:                     ";KN$
55 PRINT "{HOME}{DOWN 7}";
60 INPUT " ANFANGSKAPITAL:                 ";K0$
65 PRINT "{HOME}{DOWN 10}";
70 INPUT" ZAHL DER ZINSPERIODEN:       ";N$
80 PRINT "{HOME}{DOWN 12}";
90 INPUT" ZINSSATZ PRO PERIODE (IN %):";P$
100 PRINT"{HOME}{DOWN 17}";
105 PRINT"                                        ": REM 40 LEERZ.
110 REM   CZ=16: CS=1: GOSUB 1000
115 PRINT"                "
120 IF KN$="" THEN GOTO 210
130 IF K0$="" THEN GOTO 280
140 IF N$="" THEN GOTO 350
150 IF P$="" THEN GOTO 420
155 PRINT"{HOME}{DOWN 18}"
160 PRINT"****** FALSCHE EINGABE!"
165 FOR T=1 TO 800: NEXT T: CLR
170 GOTO 20
175 PRINT"{HOME}{DOWN 22}"
180 PRINT" ENDE: E       WEITER: RETURN"
185 GET A$: IF A$="" THEN GOTO 185
190 IF A$="E" THEN END
195 CLR: GOTO 20
```

```
200 REM.............................................BERECHNUNGEN
210 REM...............................BERECHNUNG DES ENDKAPITALS
220 K0=VAL(K0$)
230 N=VAL(N$)
240 P=VAL(P$)
250 KN=K0*(1+P/100)↑N
255 PRINT"{HOME}{DOWN}{DOWN}{DOWN}{DOWN}{DOWN}{DOWN}";TAB(30);
260 PRINT "{RIGHT}";FN R(KN)
270 GOTO 175
280 REM...........................BERECHNUNG DES ANFANGSKAPITALS
290 KN=VAL(KN$)
300 N=VAL(N$)
310 P=VAL(P$)
320 K0=KN/(1+P/100)↑N
325 PRINT"{HOME}{DOWN}{DOWN}{DOWN}{DOWN}{DOWN}{DOWN}{DOWN}{DOWN}";TAB(30);
330 PRINT "{RIGHT}";FN R(K0)
340 GOTO 175
350 REM..............................BERECHNUNG DER ZINSPERIODEN
360 K0=VAL(K0$)
370 KN=VAL(KN$)
380 P=VAL(P$)
390 N=(LOG(KN/K0))/(LOG(1+P/100))
395 PRINT"{HOME}{DOWN}{DOWN}{DOWN}{DOWN}{DOWN}{DOWN}{DOWN}{DOWN}{DOWN}{DOWN}";TAB(30);
400 PRINT "{RIGHT}"; FN R(N)
410 GOTO 175
420 REM.....................BERECHNUNG DES ZINSSATZES PRO PERIODE
430 K0=VAL(K0$)
440 KN=VAL(KN$)
450 N=VAL(N$)
460 P=100*((KN/K0)↑(1/N)-1)
465 PRINT"{HOME}{DOWN}{DOWN}{DOWN}{DOWN}{DOWN}{DOWN}{DOWN}{DOWN}{DOWN}{DOWN}{DOWN}{DOWN}";TAB(30);
470 PRINT "{RIGHT}"; FN R(P);" %"
480 GOTO 175

READY.
```

```
ZINSESZINS-BERECHNUNGEN


ENDKAPITAL:                  ? 20610.32

ANFANGSKAPITAL:              ? 10000

ZAHL DER ZINSPERIODEN:       ? 10

ZINSSATZ PRO PERIODE (IN %):? 7.5

     ENDE: E      WEITER: RETURN


ENDKAPITAL:                  ? 20000

ANFANGSKAPITAL:              ? 15485.29

ZAHL DER ZINSPERIODEN:       ? 5

ZINSSATZ PRO PERIODE (IN %):? 5.25
```

Sortier - Verfahren

Im folgenden werden vier verschiedene Sortierverfahren vorgestellt, die sich sowohl in Bezug auf den zu Grunde liegenden Algorithmus, als auch in Hinblick auf die Sortierzeiten ganz erheblich voneinander unterscheiden.

Generell kann man sagen, daß das Sortieren umso länger dauert, je einfacher die Sortiermethode ist (und umgekehrt). Die Programme sind nach zunehmender Komplexität angeordnet.

Es ist jeweils eine Version zum Sortieren von Zahlen und eine zum Sortieren von Strings angegeben.

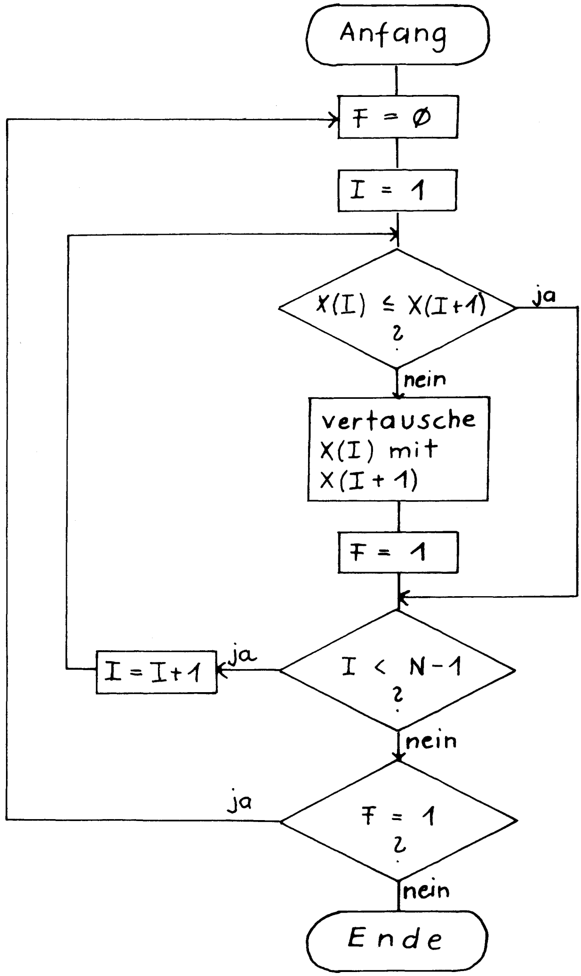
Abschließend werden alle vier Verfahren miteinander verglichen, so daß Sie sofort sehen können, welches Programm für eine bestimmte Anwendung am besten geeignet ist.

Alle Sortierprogramme wurden als Unterprogramme geschrieben, so daß sie sich leicht in eigene Programme einbauen lassen. Dies wird auch dadurch unterstützt, daß in den Unterprogrammen nur Variablen benutzt werden, deren Name mit dem Buchstaben 'X' beginnt. So ist es leicht zu vermeiden, daß die Variablen aus dem Hauptprogramm mit denen aus dem Unterprogramm in Konflikt geraten.

B u b b l e - S o r t

Der Vorteil dieses Sortierverfahrens liegt darin, daß es sehr einfach ist: die Liste der zu sortierenden Daten wird solange wiederholt durchlaufen, bis die Daten sortiert sind. Dies wird dadurch erreicht, daß jeweils zwei aufeinanderfolgende Daten miteinander verglichen werden. Stehen sie bereits in der richtigen Reihenfolge, wird der zweite Wert mit dem nachfolgenden verglichen; stehen die beiden Daten jedoch in der falschen Reihenfolge, so werden sie miteinander vertauscht. Dadurch 'sprudeln' die größeren Daten allmählich ans Ende der Liste (bubble = Blase, Sprudeln). Das Sortieren ist beendet, wenn während eines Durchgangs kein Vertauschen erfolgte.

Eine sinnvolle Anwendung dieser Sortiermethode ist allerdings auf nur sehr wenige Daten begrenzt, da die Sortierzeiten sehr schnell zunehmen. Für N Durchgänge mit je N-1 Vergleichen werden bis zu N*(N-1) Vergleiche benötigt. Sind andererseits die zu sortierenden Daten bereits vollständig sortiert, so ist das Verfahren bereits nach einem einzigen Durchlauf (also nach N-1 Vergleichen) beendet.



Fluß-Diagramm: Bubble Sort

```
10 REM...............................................BUBBLE-SORT
20 PRINT"■■ ■                BUBBELSORT                  "
30 PRINT"■■ WIEVIELE ZUFALLSZAHLEN SOLLEN SORTIERT  WERDEN?"
35 REM...............................................EINGABETEIL
40 PRINT: INPUT" ANZAHL";XN
45 DIM X(XN+1)
50 FOR I=1 TO XN
60 X(I)=RND(1)
70 NEXT I
75 TI$="000000"
80 GOSUB 1000
85 REM.........................................BILDSCHIRMAUSGABE
90 FOR I=1 TO XN
100 PRINT I;TAB(8);X(I)
110 NEXT I
120 END
1000 REM...............................BUBBLE-SORT UNTERPROGRAMM
1005 XF=0
1010 FOR XI=1 TO XN-1
1020 IF X(XI)<=X(XI+1) THEN GOTO 1070
1030 XT=X(XI)
1040 X(XI)=X(XI+1)
1050 X(XI+1)=XT
1060 XF=1
1070 NEXT XI
1080 IF XF=1 THEN GOTO 1000
1090 PRINT"■■■ SORTIERZEIT FUER";XN;"WERTE:"
1100 PRINT "■ ";MID$(TI$,3,2);" MINUTEN ";
1110 PRINT RIGHT$(TI$,2);" SEKUNDEN "
1120 PRINT"■■■ AUSGABE DER WERTE --> TASTE DRUECKEN"
1130 GET A$: IF A$="" THEN GOTO 1130
1140 PRINT
1150 RETURN

READY.
```

```
10 REM...............................BUBBLE-SORT FUER BUCHSTABEN
20 PRINT"▓▓ ▓                    BUBBELSORT               "
30 PRINT"▓▓ WIEVIELE BUCHSTABEN SOLLEN SORTIERT": PRINT" WERDEN?"
35 REM............................................EINGABETEIL
40 PRINT:INPUT" ANZAHL";XN
45 DIM X$(XN+1)
50 FOR I=1 TO XN
60 X$(I)=CHR$(RND(1)*26+65)
70 NEXT I
75 TI$="000000"
80 GOSUB 1000
85 REM........................................BILDSCHIRMAUSGABE
90 FOR I=1 TO XN
100 PRINT I;TAB(8);X$(I)
110 NEXT I
120 END
1000 REM.............................BUBBLE-SORT (UNTERPROGRAMM)
1005 XF=0
1010 FOR XI=1 TO XN-1
1020 IF X$(XI)<=X$(XI+1) THEN GOTO 1070
1030 XT$=X$(XI)
1040 X$(XI)=X$(XI+1)
1050 X$(XI+1)=XT$
1060 XF=1
1070 NEXT XI
1080 IF XF=1 THEN GOTO 1000
1090 PRINT"▓▓▓ SORTIERZEIT FUER";XN;"BUCHSTABEN:"
1100 PRINT"▓ ";MID$(TI$,3,2);" MINUTEN ";
1110 PRINT RIGHT$(TI$,2);" SEKUNDEN "
1120 PRINT"▓▓▓ AUSGABE --> TASTE DRUECKEN"
1130 GET A$: IF A$="" THEN GOTO 1130
1140 PRINT
1150 RETURN

READY.
```

Sortieren durch Einfügen

Beim Sortieren durch direktes Einfügen wird – beginnend mit dem zweiten Element (i=2) bei jedem Schritt das nächste (i-te) Element herausgegriffen und an der richtigen Stelle eingefügt (und i um 1 erhöht).

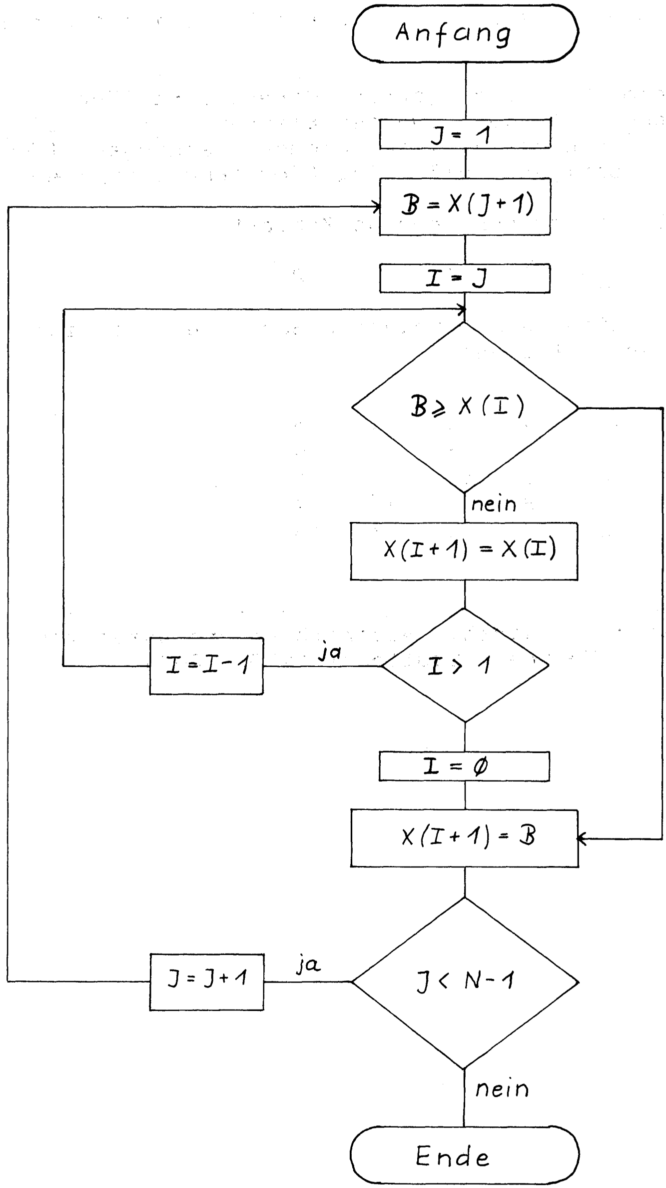
Für die zu sortierende Zahlenfolge

6 15 7 1 3 2

ergeben sich somit folgende Schritte bis zur vollständigen Sortierung:

i=2	6	15	7	1	3	2
i=3	6	7	15	1	3	2
i=4	1	6	7	15	3	2
i=5	1	3	6	7	15	2
i=6	1	2	3	6	7	15

Literatur: N. Wirth: Algorithmen und Datenstrukturen. Teubner, Stuttgart 1979, 92-94



Fluß-Diagramm: Einfügesort

```
10 REM................................SORTIEREN DURCH EINFUEGEN
20 PRINT"▒▒ ▒     SORTIEREN DURCH EINFUEGEN        "
30 PRINT"▒▒ WIEVIELE ZUFALLSZAHLEN SOLLEN SORTIERT  WERDEN? "
35 REM............................................EINGABETEIL
40 PRINT: INPUT" ANZAHL";XN
45 DIM X(XN)
50 FOR I=1 TO XN
60 X(I) =RND(1)
70 NEXT I
75 TI$="000000"
80 GOSUB 1000
85 REM........................................BILDSCHIRMAUSGABE
90 FOR I=1 TO XN
100 PRINT I;TAB(8);X(I)
110 NEXT I
120 END
1000 REM...............SORTIEREN DURCH EINFUEGEN (UNTERPROGRAMM)
1010 FOR XJ=1 TO XN-1
1020 XB=X(XJ+1)
1030 FOR XI=XJ TO 1 STEP -1
1040 IF XB>=X(XI) THEN 1080
1050 X(XI+1)=X(XI)
1060 NEXT XI
1070 XI=0
1080 X(XI+1)=XB
1090 NEXT XJ
1100 PRINT"▒▒▒ SORTIERZEIT FUER";XN;"ZUFALLSWERTE:"
1110 PRINT: PRINT" ";MID$(TI$,3,2);" MINUTEN ";
1120 PRINT RIGHT$(TI$,2);" SEKUNDEN"
1130 PRINT"▒▒▒ AUSGABE DER WERTE --> TASTE DRUECKEN"
1140 GET A$: IF A$="" THEN GOTO 1140
1150 PRINT
1160 RETURN

READY.
```

```
10 REM..................SORTIEREN DURCH EINFUEGEN FUER BUCHSTABEN
20 PRINT"□▣ ▣     SORTIEREN DURCH EINFUEGEN        "
30 PRINT"▣▣ WIEVIELE BUCHSTABEN SOLLEN SORTIERT":PRINT" WERDEN?"
35 REM....................................................EINGABETEIL
40 PRINT: INPUT" ANZAHL";XN
45 DIM X$(XN)
50 FOR I=1 TO XN
60 X$(I) =CHR$(RND(1)*26+65)
70 NEXT I
75 TI$="000000"
80 GOSUB 1000
85 REM.............................................BILDSCHIRMAUSGABE
90 FOR I=1 TO XN
100 PRINT I;TAB(8);X$(I)
110 NEXT I
120 END
1000 REM................SORTIEREN DURCH EINFUEGEN (UNTERPROGRAMM)
1010 FOR XJ=1 TO XN -1
1020 XB$=X$(XJ+1)
1030 FOR XI=XJ TO 1 STEP -1
1040 IF XB$>=X$(XI) THEN 1080
1050 X$(XI+1)=X$(XI)
1060 NEXT XI
1070 XI=0
1080 X$(XI+1)=XB$
1090 NEXT XJ
1100 PRINT"▣▣▣ SORTIERZEIT FUER";XN;"BUCHSTABEN:"
1110 PRINT: PRINT" ";MID$(TI$,3,2);" MINUTEN ";
1120 PRINT RIGHT$(TI$,2);" SEKUNDEN"
1130 PRINT"▣▣▣ AUSGABE DER WERTE --> TASTE DRUECKEN"
1140 GET A$: IF A$="" THEN GOTO 1140
1150 PRINT
1160 RETURN

READY.
```

S H E L L S O R T

SHELLSORT (von D.L.Shell) ist ein Sortierverfahren durch Einfügen mit abnehmender Schrittweite.

Sind z.B. acht Elemente zu sortieren, so werden zunächst alle Elemente, die vier Positionen voneinander entfernt sind, zusammengefaßt und getrennt sortiert (4-fach-Sortierung). Danach werden die Elemente wieder in Gruppen zusammengefaßt, die zwei Positionen voneinander entfernt sind und wiederum sortiert (2-fach-Sortierung). Schließlich werden in einem letzten Durchgang alle Elemente in einer gewöhnlichen oder 1-Sortierung geordnet.

Ist also z.B. folgende Zahlenreihe zu sortieren:

44 55 12 42 94 18 6 67

so ergibt die 4-fach-Sortierung:

44 18 6 42 94 55 12 67

die 2-fach-Sortierung ergibt:

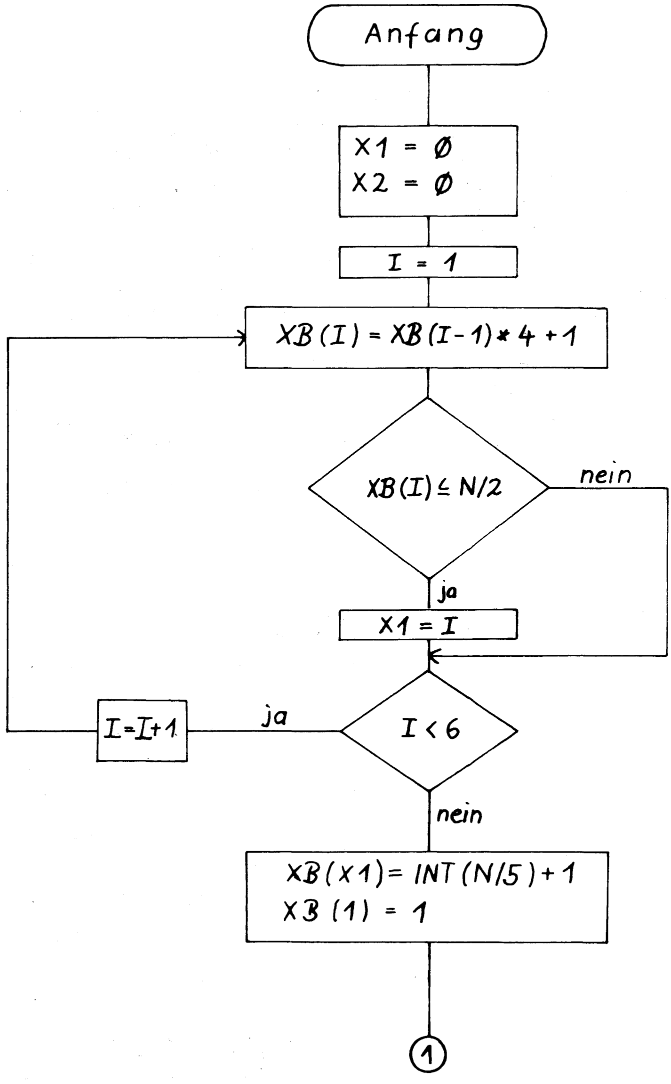
6 18 12 42 44 55 94 67

die 1-Sortierung ergibt:

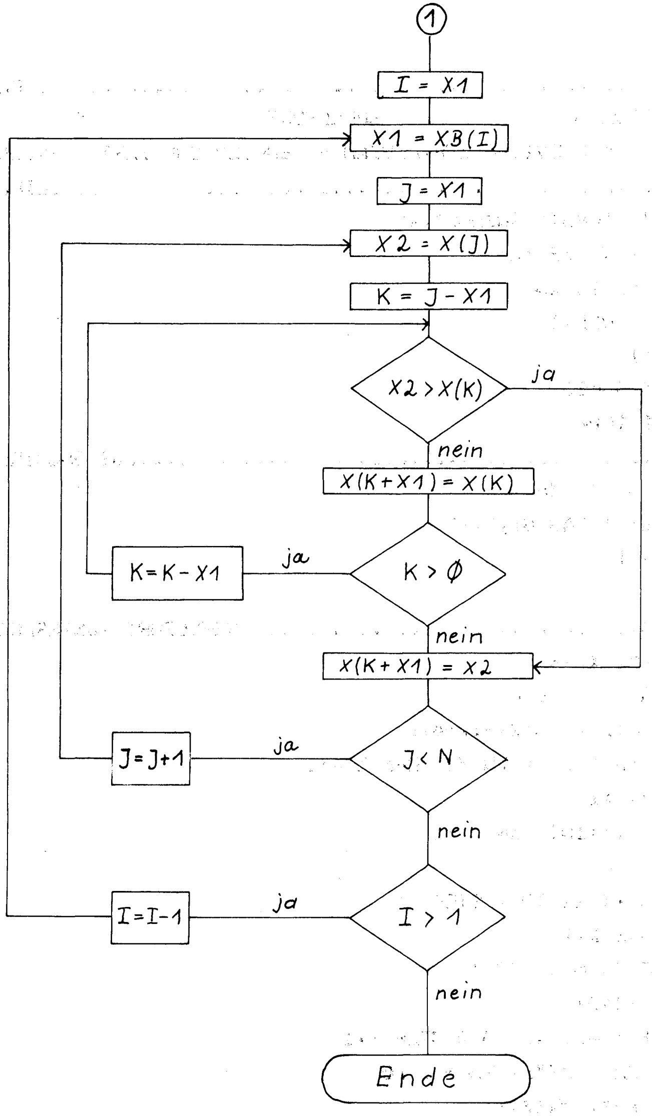
6 12 18 42 44 55 67 94

Die Methode der kleiner werdenden Schrittweiten liefert jedoch noch bessere Ergebnisse, wenn man Schrittweiten verwendet, die keine Potenzen von 2 sind.

Literatur: Shell, D.L.: A Highspeed Sorting Procedure. Comm. ACM 2, Nr. 7 (1959), 30-32



Anfang
X1 = Ø
X2 = Ø
I = 1
XB(I) = XB(I-1) * 4 + 1
XB(I) ≤ N/2
nein
ja
X1 = I
I < 6
ja
I=I+1
nein
XB(X1) = INT(N/5) + 1
XB(1) = 1
1



Fluß-Diagramm: Shellsort

```
10 REM............................................SHELLSORT
20 PRINT"                    SHELLSORT            "
30 PRINT" WIEVIELE ZUFALLSZAHLEN SOLLEN SORTIERT  WERDEN"
35 REM..........................................EINGABETEIL
40 PRINT: INPUT" ANZAHL";XN
45 DIM X(XN),XB(XN)
50 FOR I=1 TO XN
60 X(I) = RND(1)
70 NEXT I
75 TI$="000000"
80 GOSUB 1000
85 REM.....................................BILDSCHIRMAUSGABE
90 FOR I=1 TO XN
100 PRINT I;TAB(8);X(I)
110 NEXT I
120 END
1000 REM...........................SHELLSORT (UNTERPROGRAMMM)
1010 X1=0: X2=0
1020 FOR XI=1 TO 6
1030 XB(XI) = XB(XI-1)*4+1
1040 IF XB(XI) <= XN /2 THEN X1=XI
1050 NEXT XI
1060 XB(X1)=INT (XN/5)+1
1070 XB(1)=1
1080 FOR XI=X1 TO 1 STEP -1
1090 X1=XB(XI)
1100 FOR XJ = X1 TO XN
1110 X2=X(XJ)
1120 FOR XK=XJ-X1 TO 0 STEP -X1
1130 IF X2 > X(XK) THEN 1160
1140 X(XK+X1)=X(XK)
1150 NEXT XK
1160 X(XK+X1) = X2
1170 NEXT XJ
1180 NEXT XI
```

```
1190 PRINT"▒▒▒ SORTIERZEIT FUER";XN;"WERTE:"
1200 PRINT"▒ ";MID$(TI$,3,2);" MINUTEN ";
1210 PRINT RIGHT$(TI$,2);" SEKUNDEN"
1220 PRINT"▒▒▒ AUSGABE --> TASTE DRUECKEN"
1230 GET A$: IF A$="" THEN GOTO 1230
1240 PRINT
1250 RETURN

READY.
```

```
10 REM..................................SHELLSORT FUER BUCHSTABEN
20 PRINT"□▩ ▨                        SHELLSORT               "
30 PRINT"▩▩ WIEVIELE BUCHSTABEN SOLLEN SORTIERT     WERDEN"
35 REM..................................................EINGABETEIL
40 PRINT: INPUT" ANZAHL";XN
45 DIM X$(XN),XB(XN)
50 FOR I=1 TO XN
60 X$(I) = CHR$(RND(1)*26+65)
70 NEXT I
75 TI$="000000"
80 GOSUB 1000
85 REM.......................................BILDSCHIRMAUSGABE
90 FOR I=1 TO XN
100 PRINT I;TAB(8);X$(I)
110 NEXT I
120 END
1000 REM..............................SHELLSORT (UNTERPROGRAMMM)
1010 X1=0: X2$=""
1020 FOR XI=1 TO 6
1030 XB(XI) = XB(XI-1)*4+1
1040 IF XB(XI) <= XN/2 THEN X1=XI
1050 NEXT XI
1060 XB(X1)=INT(XN/5)+1
1070 XB(1)=1
1080 FOR XI=X1 TO 1 STEP -1
1090 X1=XB(XI)
1100 FOR XJ=X1 TO XN
1110 X2$=X$(XJ)
1120 FOR XK = XJ-X1 TO 0 STEP -X1
1130 IF X2$ > X$(XK) THEN 1160
1140 X$(XK+X1)=X$(XK)
1150 NEXT XK
1160 X$(XK+X1) = X2$
1170 NEXT XJ
1180 NEXT XI
```

```
1190 PRINT"QQQ SORTIERZEIT FUER";XN;"BUCHSTABEN:"
1200 PRINT"Q ";MID$(TI$,3,2);" MINUTEN ";
1210 PRINT RIGHT$(TI$,2);" SEKUNDEN"
1220 PRINT"QQQ AUSGABE --> TASTE DRUECKEN"
1230 GET A$: IF A$="" THEN GOTO 1230
1240 PRINT
1250 RETURN

READY.
```

Q U I C K S O R T

QUICKSORT (von C.A.R. Hoare) ist ein Algorithmus, der am einfachsten rekursiv formuliert werden kann:

Man wählt ein beliebiges (z.B. das mittlere) Bezugselement aus der zu sortierenden Zahlenreihe. Durch Vertauschen werden nun alle Elemente, die kleiner sind als das Bezugselement, auf die linke Seite, alle Elemente, die größer sind als das Bezugselement, auf die rechte Seite des Bezugselementes gebracht.

Gegeben sei z.B. die folgende Zahlenreihe

5 20 4 18 <u>9</u> 5 3

wobei die 9 als Bezugselement dienen soll. Man sucht nun von links aus durch das Feld (Index XI) nach einem Element, das größer ist als 9: in unserem Fall die 20. Dann durchsucht man das Feld von rechts aus (Index XJ) nach einem Element, das kleiner ist als das Bezugselement, hier also die 3. Diese beiden Elemente werden im nächsten Schritt vertauscht:

5 3 4 18 <u>9</u> 5 20

Die Suche wird dann im Bereich der ausgetauschten Elemente fortgesetzt, sodaß beim nächsten Schritt die 18 und die 5 vertauscht werden:

5 3 4 5 <u>9</u> 18 20

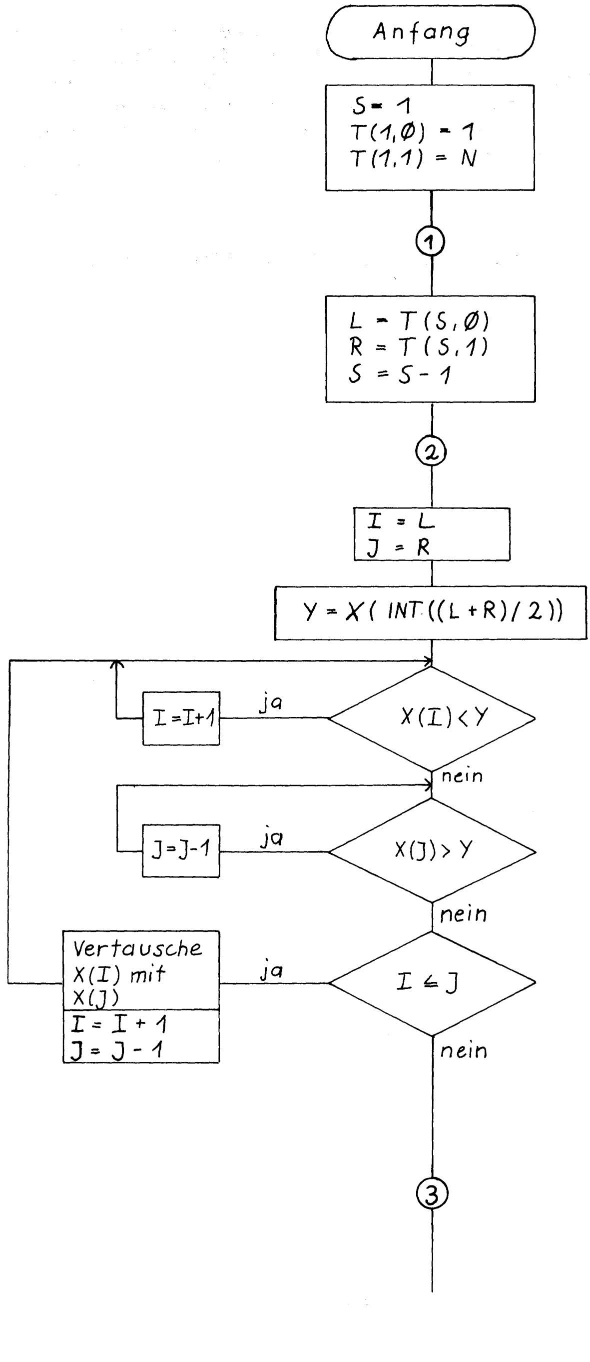
Würde man so weitermachen, würden im nächsten Schritt die 18 und die 5 wieder vertauscht: das Verfahren muß also abgebrochen werden, wenn man bei der Suche von links und rechts auf das gleiche Feldelement stößt.

Damit stehen nun links vom Bezugselement nur kleinere und rechts nur größere Feldelemente.

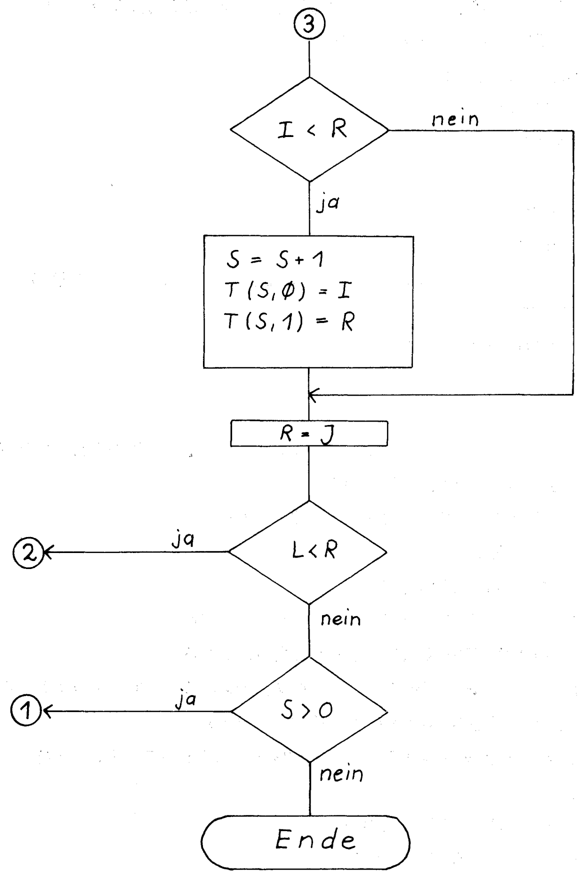
Auf die so entstandenen beiden Teilfelder wendet man wieder das gleiche Verfahren an, und zwar solange, bis jedes Teilfeld nur noch aus einem Element besteht. Damit ist dann das ganze Feld sortiert.

Da eine rekursive Formulierung in BASIC nicht möglich ist, benötigt man hier ein Hilfsfeld, das als Stapelregister (Stack) dient. Darin wird notiert, welcher Teil des zu sortierenden Feldes noch bearbeitet werden muß. (Für die Dimension dieses Feldes ergaben sich beim Sortieren von bis zu 1000 Zufallszahlen Werte zwischen 9 und 13).

Literatur: Hoare, C.A.R.: QUICKSORT. Comp. J. 5, Nr. 1 (1962), 10-15



Anfang
S = 1
T(1,Ø) = 1
T(1,1) = N
1
L = T(S,Ø)
R = T(S,1)
S = S - 1
2
I = L
J = R
Y = X(INT((L+R)/2))
X(I) < Y
ja
I = I+1
nein
X(J) > Y
ja
J = J-1
nein
I ≤ J
ja
Vertausche X(I) mit X(J)
I = I + 1
J = J - 1
nein
3



Fluß-Diagramm: Quicksort

```
10 REM...................................................QUICKSORT
20 PRINT"▒▒ ▒                 QUICKSORT             "
30 PRINT"▒▒ WIEVIELE ZUFALLSZAHLEN SOLLEN SORTIERT"
40 PRINT"▒ WERDEN";: INPUT XN
50 DIM X(XN),XT(20,1)
60 FOR I=1 TO XN
70 X(I)=RND(1)
80 NEXT I
90 TI$="000000"
100 GOSUB 1000
110 REM.........................................BILDSCHIRMAUSGABE
120 FOR I=1 TO XN
130 PRINT I;TAB(8);X(I)
140 NEXT I
150 END
1000 REM.................................QUICKSORT (UNTERPROGRAMM)
1010 XS=1: XT(1,0)=1: XT(1,1)=XN
1020 XL=XT(XS,0): XR=XT(XS,1): XS=XS-1
1030 XI=XL: XJ=XR
1040 XY=X(INT((XL+XR)/2))
1050 IF X(XI)<XY THEN XI=XI+1: GOTO 1050
1060 IF X(XJ)>XY THEN XJ=XJ-1: GOTO 1060
1070 IF XI<=XJ THEN XW=X(XI): X(XI)=X(XJ): X(XJ)=XW: XI=XI+1
1080 IF XI<=XJ THEN XJ=XJ-1: GOTO 1050
1090 IF XI<XR THEN XS=XS+1: XT(XS,0)=XI: XT(XS,1)=XR
1100 XR=XJ
1110 IF XL<XR THEN GOTO 1030
1120 IF XS>0 THEN GOTO 1020
1130 PRINT"▒▒ SORTIERZEIT FUER";XN;"ZUFALLSZAHLEN:"
1140 PRINT" ";MID$(TI$,3,2);" MINUTEN ";RIGHT$(TI$,2);" SEKUNDEN"
1150 PRINT "▒▒ AUSGABE DER ZAHLEN -> TASTE DRUECKEN"
1160 GET Z$: IF Z$="" THEN GOTO 1160
1170 PRINT
1180 RETURN

READY.
```

```
10 REM...............................QUICKSORT FUER BUCHSTABEN
20 PRINT"░░ ░            QUICKSORT            "
30 PRINT"░░ WIEVIELE BUCHSTABEN SOLLEN SORTIERT"
40 PRINT"░ WERDEN";: INPUT XN
50 DIM X$(XN),XT(20,1)
60 FOR I=1 TO XN
70 X$(I)=CHR$(RND(1)*26+65)
80 NEXT I
90 TI$="000000"
100 GOSUB 1000
110 REM........................AUSGABE DER SORTIERTEN BUCHSTABEN
120 FOR I=1 TO XN
130 PRINT I;TAB(8);X$(I)
140 NEXT I
150 END
1000 REM...............................QUICKSORT-UNTERPROGRAMM
1010 XS=1: XT(1,0)=1: XT(1,1)=XN
1020 XL=XT(XS,0): XR=XT(XS,1): XS=XS-1
1030 XI=XL: XJ=XR
1040 XY$=X$(INT((XL+XR)/2))
1050 IF X$(XI)<XY$ THEN XI=XI+1: GOTO 1050
1060 IF X$(XJ)>XY$ THEN XJ=XJ-1: GOTO 1060
1070 IF XI<=XJ THEN XW$=X$(XI): X$(XI)=X$(XJ): X$(XJ)=XW$:XI=XI+1
1080 IF XI<=XJ THEN XJ=XJ-1: GOTO 1050
1090 IF XI<XR THEN XS=XS+1: XT(XS,0)=XI: XT(XS,1)=XR
1100 XR=XJ
1110 IF XL<XR THEN GOTO 1030
1120 IF XS>0 THEN GOTO 1020
1130 PRINT"░░ SORTIERZEIT FUER";XN;"BUCHSTABEN:"
1140 PRINT" ";MID$(TI$,3,2);" MINUTEN ";RIGHT$(TI$,2);" SEKUNDEN"
1150 PRINT "░░ AUSGABE -> TASTE DRUECKEN"
1160 GET Z$: IF Z$="" THEN GOTO 1160
1170 PRINT
1180 RETURN

READY.
```

Vergleich der Sortierzeiten für N Zufallszahlen

(in Sekunden)

N	Bubblesort	Einfügesort	Shellsort	Quicksort
5	0.5	0.6	0.8	0.8
10	1.0	0.8	0.9	1.0
20	5.3	1.5	1.7	2.0
50	38	10.5	6.7	7.5
100	118	39	14	17.5
200	632	148	36.5	39.5
500	-	947	112	112
1000	-	-	262	252

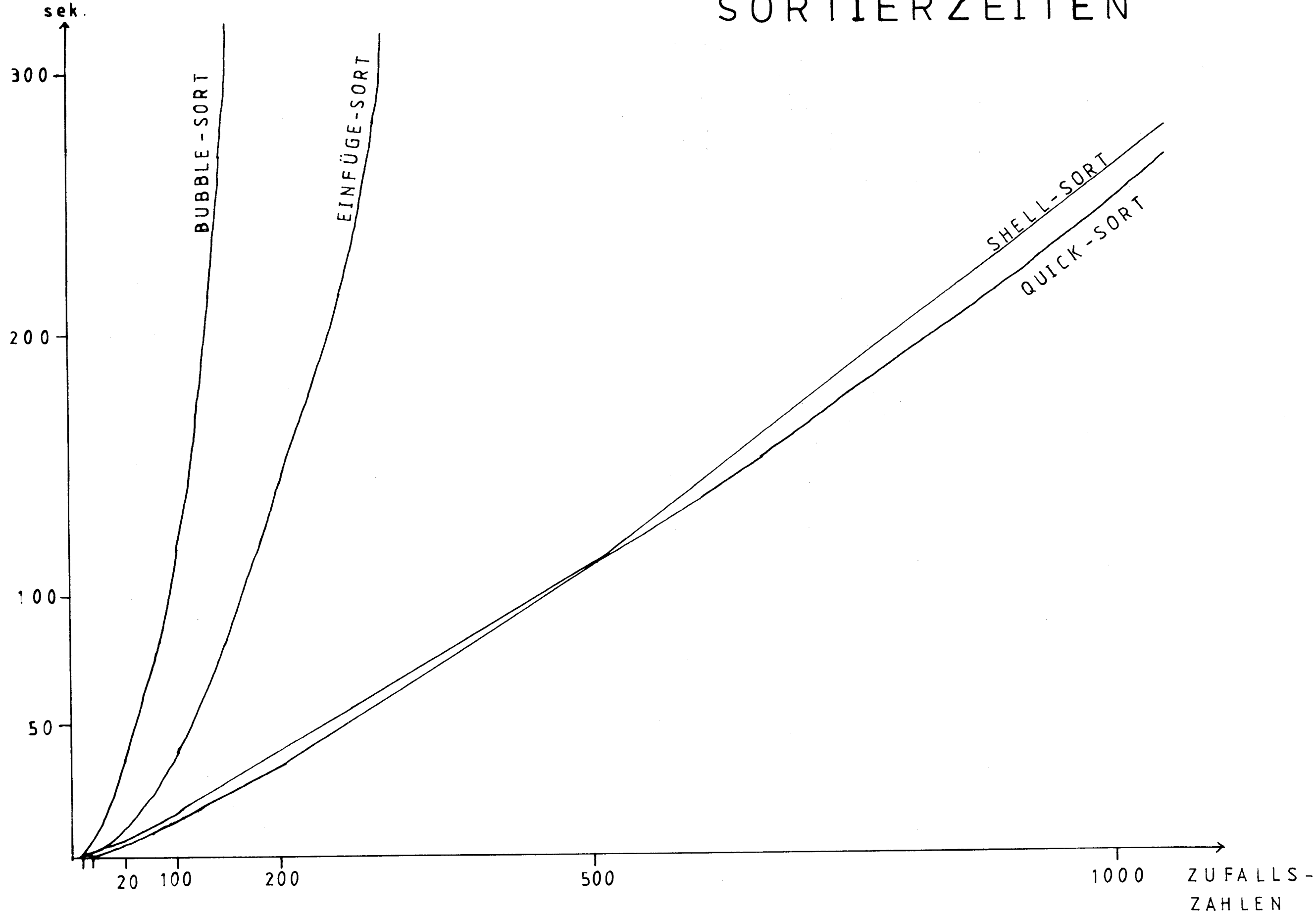
Die Daten aus dieser Tabelle sind in der folgenden Abbildung auch noch grafisch dargestellt. Die jeweils kürzesten Sortierzeiten für einen bestimmten Wert von N sind in der Tabelle markiert.

Wie Sie sehen, ist das einfache Bubblesort das mit Abstand schlechteste Sortierverfahren, wenn mehr als 10 Werte zu sortieren sind. Andererseits ist es für sehr wenige Daten sogar noch etwas günstiger als die komplizierteren Algorithmen.

Für bis zu etwa 20 Daten ist das Sortieren durch Einfügen am besten geeignet und bietet auch noch für etwa bis zu 50 Daten durchaus akzeptable Sortierzeiten. Sind die Daten, die den Sortierprogrammen übergeben werden, bereits vorher sortiert, so wird dies vom Einfüge-Sortieren übrigens am schnellsten "erkannt"; in solchen Fällen ist es den beiden kompexeren Verfahren Shellsort und Quicksort weit überlegen.

Sind etwa 50 bis 200 Daten zu sortieren, ist das Shellsort-Verfahren am schnellsten, wenngleich der Unterschied zum Quicksort nur gering ist. Die Differenz zwischen diesen beiden Algorithmen vergrößert sich jedoch allmählich, wenn die Zahl der zu sortierenden Daten zunimmt: bei mehr als 500 Daten ist Quicksort das schnellste Sortierverfahren.

SORTIERZEITEN



1 7 u n d 4

Mit diesem Programm wird das Kartenspiel 17 und 4 nachgebildet. Abwechselnd "ziehen" der Spieler und der Computer jeweils eine Karte. Wer insgesamt mehr als 21 Augenzahlen erreicht, hat verloren, sonst gewinnt die höhere Augenzahl.

Zum "Ziehen" der Karten werden mit der Funktion RND Zufallszahlen im Bereich von 2 bis 4 und von 7 bis 11 erzeugt. Die Strategie des Computers besteht darin, nur dann noch eine weitere Karte zu nehmen, wenn die Summe der Augenzahlen nicht größer als 15 + 1 ist. Auch diese Entscheidung erfolgt wieder zufallsbedingt.

```
10 REM........................................SIEBZEHN UND VIER
20 PRINT"□▥ ▨          SIEBZEHN UND VIER          "
30 PRINT:PRINT SPC(9);"(SPIEL BEENDEN MIT: E)"
35 REM............................................KARTEN ZIEHEN
40 S1=0: C1=0
50 S=0: C=0: E=1: S$="J":A$=""
60 Z=0
70 IF S>=21 THEN A$="N"
80 IF A$="N" THEN 220
85 CZ=6: CS=1: GOSUB 750
90 PRINT"WILLST DU EINE KARTE? (J/N)";: GET A$
95 IF A$="" THEN GOTO 85
100 CZ=12: CS=1: GOSUB 750
110 IF A$="E" THEN END
120 IF A$="N" THEN GOTO 220
130 X=INT(10*RND(1))+2
140 IF X>4 AND X<7 THEN GOTO 130
150 IF Z=1 THEN GOTO 430
155 REM.....................................AUGENZAHL AUSGEBEN
160 S=S+X
170 CZ=8: CS=15: GOSUB 750
175 PRINT"AUGENZAHL  -->       ";
190 IF X>21 THEN PRINT"▨";
200 CZ=8: CS=30: GOSUB 750: PRINT S;
220 Z=1
230 IF E=1 THEN GOTO 130
240 IF A$="J" THEN GOTO 60
250 CZ=14: CS=1: GOSUB 750
255 PRINT"ICH HABE"C"AUGEN: ";
260 IF C>21 AND S>21 THEN 380
270 IF C<>S THEN 300
```

```
275 REM..................................ANGABE DES GEWINNERS
280 CZ=15:CS=1: GOSUB 750
285 PRINT"*** UNENTSCHIEDEN!                 ";
290 GOTO 390
300 IF C>21 THEN GOTO 315
310 IF S>21 OR C>S THEN GOTO 350
315 CZ=15: CS=1: GOSUB 750
320 PRINT">>> DU HAST GEWONNEN!!          ";
330 S1=S1+1
340 GOTO 390
350 CZ=15: CS=1: GOSUB 750
355 PRINT"... DIESMAL HABE ICH GEWONNEN!";
360 C1=C1+1
370 GOTO 390
380 CZ=15: CS=1: GOSUB 750
385 PRINT"DAS WAR FUER UNS BEIDE NICHTS ...";
390 CZ=18: CS=1: GOSUB 750: PRINT" NEUES SPIEL? (J/N)";
395 GET SP$: IF SP$="" THEN GOTO 395
396 REM...............................ANGABE DES SPIELSTANDES
400 PRINT"□": CZ=1: CS=1: GOSUB 750: PRINT"SPIELSTAND:"
405 CZ=3: CS=1: GOSUB 750:PRINT" DU GEGEN MICH ";
410 PRINT"▣";S1;"▌▌ :";C1;"▌▌ "
420 IF SP$="N" THEN PRINT"▒▒▒":END
425 CZ=15: CS=1: GOTO 50
426 REM.........................................COMPUTERZUG
430 IF C>14+INT(3*RND(1)) THEN GOTO 470
440 C=C+X
450 CZ=12: CS=1: GOSUB 750: PRINT"ICH HABE GEZOGEN ...     ";
460 GOTO 60
470 E=0
480 CZ=12: CS=1: GOSUB 750: PRINT"...ICH ZIEHE NICHT MEHR!";
490 GOTO 60
```

```
750 REM............................................CURSORPOSITION
760 CZ=1024+40*CZ
770 POKE 209,CZAND255
780 POKE 210,CZ/256
790 POKE 211,CS
800 RETURN

READY.
```

```
                 SIEBZEHN UND VIER

               (SPIEL BEENDEN MIT: E)


        WILLST DU EINE KARTE? (J/N)

                      AUGENZAHL  -->  18



        ...ICH ZIEHE NICHT MEHR!

        ICH HABE 23 AUGEN:
        >>> DU HAST GEWONNEN!!


         NEUES SPIEL? (J/N)
```

N I M - S p i e l

Zwei Spieler nehmen abwechselnd von einem Haufen Streichhölzer jeweils maximal z.B. 3, mindestens jedoch 1 Streichholz weg. Wer das letzte Streichholz wegnehmen kann, gewinnt.

Vor Spielbeginn wird die Gesamtzahl der Hölzer sowie das Maximum pro Zug angegeben. Anschließend nehmen der Computer und der Spieler abwechselnd eine beliebige Zahl von Hölzern - innerhalb der zulässigen Grenzen natürlich!

Falls Sie das Spiel nicht kennen, werden Sie feststellen, daß Sie regelmäßig verlieren. Es gibt nämlich eine eindeutige Gewinnstrategie, durch die der Sieger schon vom ersten Zug an feststeht, wenn die Strategie durchgehalten wird! - Es soll hier nur ein Hinweis auf die Spielstrategie gegeben werden: versuchen Sie herauszubekommen, welche Positionen (Hölzchenzahlen) Verluststellungen darstellen. Im übrigen steht die Strategie ja auch noch (fast) im Klartext im Programm.

```
10 REM.......................................EINFACHES NIM-SPIEL
20 PRINT"☐▧ ▧          EINFACHES NIM-SPIEL        "
30 PRINT"▧▧ VON EINEM HAUFEN HOELZER NEHMEN ZWEI"
40 PRINT"▧ SPIELER ABWECHSELND MINDESTENS JE EINS,"
50 PRINT" MAXIMAL JEDOCH Z.B. DREI STUECK."
60 PRINT"▧ WER DEN REST WEGNIMMT, HAT GEWONNEN!"
70 INPUT"▧▧ WIEVIELE HOELZER";N: PRINT
80 INPUT" MAXIMUM PRO ZUG";M
90 PRINT"☐▧----------------------------------------"
95 IF A$<>"" THEN PRINT"☐ NEUES SPIEL MIT";N;"HOELZERN:"
99 REM.............................................COMPUTERZUG
100 NN=N
110 X=N/(M+1)
120 IF X=INT(X) THEN GOTO 140
130 N=(M+1)*INT(X)+1
140 N=N-1
150 PRINT"▧▧▧▧▧";TAB(18);"ICH REDUZIERE AUF";N;"▮  "
160 IF N>0 THEN GOTO 200
170 PRINT"▧▧▧▧▧▧ ICH HABE GEWONNEN! - WEITER (J/N)";:INPUT A$
180 IF A$="N" THEN END
190 GOTO 70
199 REM..............................................SPIELERZUG
200 PRINT"▧▧▧▧▧▧▧▧ WIEVIEL NIMMST DU";
210 INPUT Z
220 IF Z<1 OR Z>M THEN 200
230 N=N-Z
240 PRINT
250 PRINT"▧▧▧▧▧▧▧▧▧▧▧ DU REDUZIERST AUF";N;"▮  "
260 PRINT
270 IF N>0 THEN GOTO 110
280 PRINT"▧▧▧▧▧▧ DU HAST GEWONNEN! - WEITER (J/N)";: INPUT A$
290 IF A$="N" THEN END
300 N=INT(RND(1)*NN+10)
310 NN=N
```

```
320 PRINT"⌝▧----------------------------------------"
330 PRINT"⌝ NEUES SPIEL MIT";N;"HOELZERN"
340 GOTO 200
350 END

READY.
```

```
                  EINFACHES NIM-SPIEL

        VON EINEM HAUFEN HOELZER NEHMEN ZWEI

        SPIELER ABWECHSELND MINDESTENS JE EINS,

        MAXIMAL JEDOCH Z.B. DREI STUECK.

        WER DEN REST WEGNIMMT, HAT GEWONNEN!

        WIEVIELE HOELZER? 15

        MAXIMUM PRO ZUG? 3
```

```
----------------------------------------

                 ICH REDUZIERE AUF 0

 WIEVIEL NIMMST DU? 3

 DU REDUZIERST AUF 1

 ICH HABE GEWONNEN! - WEITER (J/N)? N

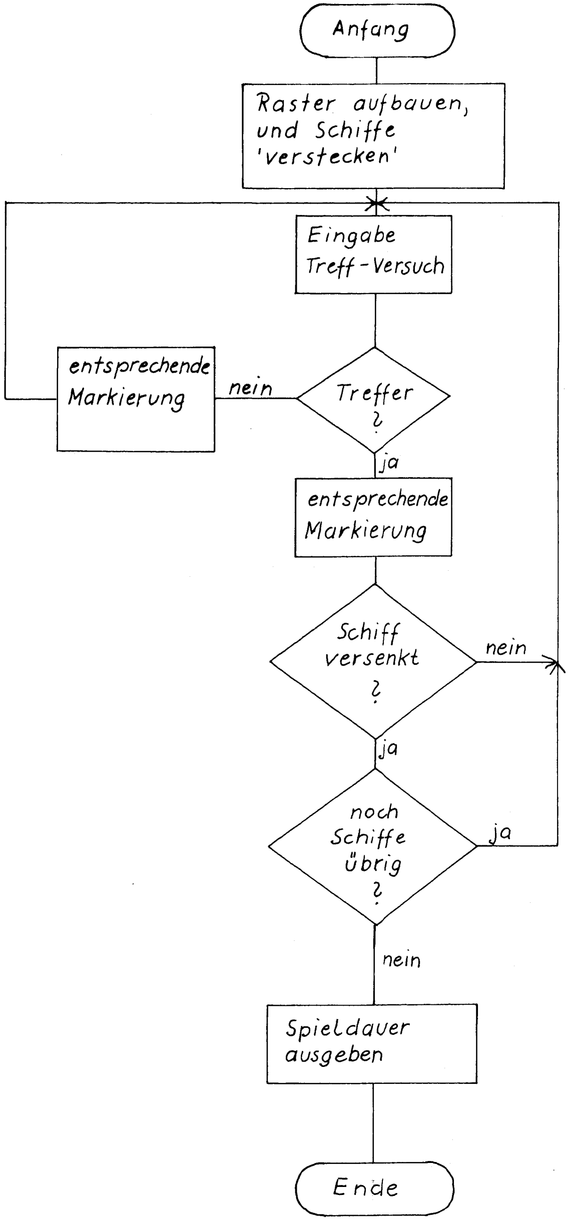
READY.
```

Schiffe versenken

Wenn Sie einmal keinen Spielpartner haben, mit dem Sie auf dem Papier "Schiffe versenken" können, kann der Computer Ihren Partner vertreten.

Die Regeln für dieses Spiels sind Ihnen sicherlich bekannt: In einem quadratischen Gitter aus NxN Kästchen sind "Schiffe" unterschiedlicher Größe zufällig verteilt worden. Durch Angabe der Koordinaten sollen die Schiffe "versenkt" werden. In dem entsprechenden Kästchen wird jeweils markiert, ob es sich um einen Treffer handelte, oder ob ins Wasser gezielt wurde. Bei einem Treffer erscheint eine Zahl von 1 bis 4, die angibt, wie lang das getroffene Schiff ist.

Die Schiffe werden mit Hilfe der Zufallsfunktion RND in dem vorgegebenen Spielfeld verteilt. Ihre Lage kann sowohl horizontal als auch vertikal oder diagonal sein. Es ist nicht ausgeschlossen, daß sich Schiffe berühren oder sogar diagonal überschneiden. (Falls Ihnen das zu unrealistisch ist, haben Sie damit gleich eine Anregung zur Änderung des Programms!)



Fluß-Diagramm: Schiffe versenken

```
10 REM.......................................SCHIFFE VERSENKEN
20 PRINT"▯▯ ▯           SCHIFFE VERSENKEN         "
240 DIM F(20,20)
250 F$=".4321": REM WASSER/SCHIFFLAENGE=4
260 PRINT"▯▯ IN EINEM QUADRAT MIT DER KANTENLAENGE"
280 PRINT" N SIND FOLGENDE SCHIFFE VERSTECKT:"
290 PRINT: PRINT" 2 MIT DER LAENGE 4"
300 PRINT" 3 MIT DER LAENGE 3"
310 PRINT" 4 MIT DER LAENGE 2"
320 PRINT" 5 MIT DER LAENGE 1"
330 PRINT: PRINT" DIE SCHIFFE KOENNEN SICH DIAGONAL"
340 PRINT" UEBERSCHNEIDEN ODER SICH BERUEHREN.▯▯▯▯"
350 INPUT" FELDGROESSE N (10 <= N <= 18)";N
360 IF N<10 OR N>18 THEN PRINT"▯▯": GOTO 350
370 REM.....................................AUSGABE DES FELDES
380 FOR I=1 TO N
390 READ A$
400 B$=B$+A$
410 NEXT I
420 PRINT"▯▯▯▯▯▯";B$
430 RESTORE
440 FOR I=1 TO N
450 READ A$
460 PRINT" ";A$
470 NEXT I
480 REM..............................'VERSTECKEN' DER SCHIFFE
490 FOR F=2 TO 5
500 FOR I=1 TO F
510 X=INT((N-6)*RND(1))+6-F
520 Y=INT((N-6)*RND(1))+6-F
530 A=INT(3*RND(1))-1
540 B=INT(3*RND(1))-1
550 IF A=0 AND B=0 THEN GOTO 540
```

```
560 FOR K=1 TO 6-F
570 IF F(X,Y)<>0 THEN GOTO 510
580 X(K)=X: Y(K)=Y
590 X=X+A: Y=Y+B
600 NEXT K
610 FOR K=1 TO 6-F
620 F(X(K),Y(K))=-F
630 NEXT K
640 NEXT I
650 NEXT F
660 REM..................................EINGABE DER KOORDINATEN
670 PRINT"▒";: FOR R=1 TO 22: PRINT"▒";: NEXT R
680 INPUT" KOORDINATEN X,Y";X$,Y$
690 IF ASC(X$)>ASC(A$) OR ASC(Y$)>ASC(A$) THEN GOTO 670
700 X=ASC(X$)-64: Y=ASC(Y$)-64
710 IF F(X,Y)>0 THEN GOTO 670
720 PRINT"▒▒";
730 Z=F(X,Y)+6
740 IF Z<6 THEN 800
750 REM...........................AUSGABE DES GETROFFENEN ZIELES
760 PRINT"                                        "
770 PRINT"┐ WASSER! BEI ";CHR$(X+64);",";CHR$(Y+64)
780 F(X,Y)=1
790 GOTO 840
800 PRINT"                                        "
810 PRINT"┐ TREFFER ";MID$(F$,6-Z,1);" BEI ";CHR$(X+64);",";
820 PRINT CHR$(Y+64)
830 F(X,Y)=ABS(F(X,Y))
840 E=0
850 PRINT"▒▒▒▒▒";
860 FOR I=1 TO N
870 FOR K=1 TO N
880 X=F(I,K)
```

```
890 IF X<=0 THEN GOTO 930
900 PRINT TAB(K+1);MID$(F$,X,1);
910 IF X=1 THEN GOTO 930
920 E=E+1
930 NEXT K
940 PRINT
950 NEXT I
960 IF E<30 THEN GOTO 670
970 DATA A,B,C,D,E,F,G,H,I,J,K,L,M,N,O,P,Q,R
980 PRINT"{CLR}";: FOR R=1 TO 22: PRINT"{DOWN}";:NEXT R
990 PRINT" ALLES VERSENKT!                    "
1000 END

READY.
```

```
  A B C D E F G H I J K L M N O P Q R
A .     . . . . . . . . . . . . . . .
B . .   . .       . . . . . 2 .
C   . . . . . . .   .   1 2 .
D . . . 1 . . . . . .     .
E . . 1 . . 1 . . . .   .   .
F . . . . . .   4 . . . .
G   . .     . . 4 . .   .   .   .
H . . . . . . . 4 . 3 .   .   .
I       . . 2   4 3 . . .   .
J . . . . . 2 . 3 4 .     .
K   . . . . . . . 4 2 2 2 . .
L . . . . 1 .   . 4 2   .     .
M . . 3 3 3 . . . 4 .   .   .
N   . 3 . . .   . . .   . .
O . . 3 . . . . . . . .   .   .
P . . 3 .   . . . . . .   .   .
Q     . . .   .   .
R   . . . .       .
ALLES VERSENKT!

READY.
```

R o u l e t t e

Bevor Sie Ihr Geld ins Spielkasino tragen, können Sie mit diesem Programm die "todsicheren" Spiel-Systeme erst einmal "auf dem Trockenen" ausprobieren - oder auch einfach nur zum Spaß den Computer als Roulette benutzen. Einen Spielplan müssen Sie sich allerdings bei Bedarf selbst malen!

Nach Drücken einer Taste wird mit der RND-Funktion zufällig eine der Zahlen von 0 bis 36 erzeugt. Zusätzlich erfolgt vom Programm aus eine Zuordnung zu den Chancen:

zweifach: Rot/Schwarz
Pair/Impair
Manque/Passe

dreifach: 1./2./3. Dutzend
1./2./3. Reihe

Anregung: Das Programm könnte so ausgebaut werden, daß auch die Spieleinsätze vom Computer verwaltet werden, sodaß gar kein realer Roulette-Spielplan benötigt wird.

```
10 REM.................................................ROULETTE
20 PRINT"▒▓ ▚            R O U L E T T E          "
30 PRINT"▓▓▓▓▓  FAITES VOTRE JEUX!..."
40 PRINT: PRINT TAB(10)"<TASTE DRUECKEN>"
50 GET N$: IF N$="" THEN GOTO 50
60 PRINT"▒": POKE 53281,3: POKE 53280,5: PRINT "▝";
70 S=17
80 PRINT"▓ DIE KUGEL ROLLT"
90 FOR K=1 TO RND(1)*110+1
100 PRINT"▚▓▓▓▓▓▓▓▓▓▓▓▓▚";TAB(22);INT(RND(1)*37);"▌ ■ ";
120 PRINT"▚▓▓";TAB(S);"."
140 S=S+1
150 NEXT K
160 PRINT
170 X=INT(37*RND(1))
180 PRINT"▚▓▓▓▓▓▓▓▓▓▓▓ DIE KUGEL FAELLT AUF ▚";X;"▌ ■ ▚";
190 IF X=0 THEN GOTO 480
200 IF X=1 OR X=3 OR X=5 THEN GOTO 420
210 IF X=7 OR X=9 OR X=12 THEN GOTO 420
220 IF X=14 OR X=16 OR X=18 THEN GOTO 420
230 IF X=19 OR X=21 OR X=23 THEN GOTO 420
240 IF X=25 OR X=27 OR X=30 THEN GOTO 420
250 IF X=32 OR X=34 OR X=36 THEN GOTO 420
260 PRINT TAB(27);"■ SCHWARZ ▝": PRINT
270 IF X=2*INT(X/2) THEN GOTO 440
280 PRINT TAB(27);"IMPAIR"
290 IF X>18 THEN GOTO 460
300 PRINT TAB(27)"MANQUE"
310 Y=INT((X-1)/12)+1
320 PRINT TAB(26);Y;"▌. DUTZEND"
330 X=X-3*INT(X/3)
340 IF X>0 THEN 360
350 X=3
360 PRINT TAB(26);X;"▌. REIHE"
370 PRINT"▓▓▓▓ BITTE AUSZAEHLEN UND SETZEN!"
380 PRINT"▚▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓": PRINT" <WEITER: TASTE, ENDE: E>"
```

```
390 GET A$: IF A$="" THEN GOTO 390
395 IF A$="E" THEN END
400 PRINT"□";
410 GOTO 60
420 PRINT TAB(27);"▧ ROT ▨": PRINT
430 GOTO 270
440 PRINT TAB(27);"PAIR"
450 GOTO 290
460 PRINT TAB(27);"PASSE"
470 GOTO 310
480 PRINT TAB(27);"ZERO!!!"
490 PRINT"▧▧ EINFACHE CHANCEN BLEIBEN,"
500 PRINT"▧ DER REST AN DIE BANK!!"
510 GOTO 380

READY.
```

```
                              DIE KUGEL ROLLT ....................
                             ..................................



                              DIE KUGEL FAELLT AUF  31   SCHWARZ

                                                        IMPAIR
                                                        PASSE
                                                        3. DUTZEND
                                                        1. REIHE



                              BITTE AUSZAEHLEN UND SETZEN!

                              <WEITER: TASTE, ENDE: E>
```

```
DIE KUGEL ROLLT .......................
......................................




DIE KUGEL FAELLT AUF  0   ZERO!!!

EINFACHE CHANCEN BLEIBEN,

DER REST AN DIE BANK!!




<WEITER: TASTE, ENDE: E>
```

```
DIE KUGEL ROLLT .....




DIE KUGEL FAELLT AUF  12   ROT
                          PAIR
                          MANQUE
                          1. DUTZEND
                          3. REIHE



BITTE AUSZAEHLEN UND SETZEN!

<WEITER: TASTE, ENDE: E>
```

Wörterraten

Bei diesem Spiel geht es darum, mit möglichst wenigen Versuchen ein bestimmtes Wort zu erraten.

Die einzige Information, die man zunächst besitzt, ist die Länge des gesuchten Wortes, die durch eine entsprechende Anzahl von Strichen dargestellt wird.

Tippt man nun eine Buchstabentaste, so wird dieser Buchstabe sofort an allen Stellen im Wort angezeigt, an denen er vorkommt - sofern er überhaupt in dem Wort vorkommt!

Hat man das Wort vollständig erraten, so wird die Anzahl der Versuche absolut und in Prozent angegeben. Hatte das gesuchte Wort z.B. 7 Buchstaben und wurden zum Erraten 14 Versuche benötigt, beträgt also die absolute Zahl der Versuche 14 und die prozentuale 14/7*100 = 200%. Die relative Zahl der Versuche kann natürlich auch unter 100% liegen, nämlich dann, wenn durch einen Versuch ein Buchstabe getroffen wurde, der an mehreren Stellen im zu erratenden Wort vorkommt.

Die zu erratenden Wörter stehen im Programm in DATA-Zeilen. Als Beispiel haben wir Namen von Personen aus Geschichten von Wilhelm Busch gewählt.

Max und Moritz

```
10 REM.........................................BEGRIFFE RATEN
15 PRINT"⊐⊠ ⊠      *****  BEGRIFFE RATEN  *****      "
20 PRINT:PRINT" WILLKOMMEN ZUM SPIEL 'BEGRIFFE RATEN'"
30 PRINT"⊠⊠ IHRE AUFGABE IST ES, EINEN VOM COMPUTER";
35 PRINT" GEWAEHLTEN BEGRIFF MIT MOEGLICHST "
40 PRINT" WENIGEN VERSUCHEN ZU ERRATEN."
45 PRINT"⊠ DIE ZAHL DER VERSUCHE";
50 PRINT" ERGIBT SICH AUS   DER ANZAHL DER STRICHE."
55 PRINT"⊠⊠⊠           *** VIEL ERFOLG ***"
60 FOR P=0 TO 6000: NEXT P
75 REM.....................................EINLESEN DER DATEN
80 RESTORE: I=INT(18*RND(1))
85 FOR A=1 TO I: READ D$: NEXT A
90 PRINT"⊐": CZ=15: CS=0: GOSUB 50000
100 FOR Y=1 TO LEN (D$)
110 PRINT" -";: NEXT Y
120 V=0: PRINT: PRINT: D=0
125 CZ=23: CS=0: GOSUB 50000
130 PRINT"BITTE GEBEN SIE IHRE BUCHSTABEN EIN": Y=1
135 GET C$: IF C$="" THEN GOTO 135
140 V=V+1
150 IF MID$(D$,Y,1)= C$ THEN GOTO 200
160 IF Y<=LEN(D$) THEN GOTO 180
170 Y=1: GOTO 135
180 Y=Y+1: GOTO 150
200 CZ=13: CS=2*Y-1: GOSUB 50000: PRINT C$: D=D+1
230 IF D=LEN(D$) THEN GOTO 280
240 IF Y<LEN(D$) THEN Y=Y+1: GOTO 150
250 GOTO 170
280 FOR P=1 TO 1000: NEXT P
```

```
300 PRINT"▯▩ ▧ ***** HERZLICHEN GLUECKWUNSCH! ***** ": PRINT
310 PRINT" SIE HABEN DEN 'UNBEKANNTEN' BEGRIFF"
315 PRINT"▩■■■■■■■■■■▧";D$: PRINT
320 PRINT" MIT";V;"VERSUCHEN ERRATEN!": PRINT"▩ DAS SIND";
325 PRINT INT(V/LEN(D$)*10000+.5)/100;"% DER LAENGE DES"
326 PRINT" BEGRIFFES."
330 PRINT"▩▨▩": INPUT" NEUES SPIEL GEFAELLIG (J/N)";A$
340 IF A$= "N" THEN END
350 GOTO 80
400 DATA MAX UND MORITZ,WITWE BOLTE,SCHNEIDER BOECK
405 DATA LEHRER LAEMPEL,ONKEL FRITZ,MEISTER BAECKER,BAUER MECKE
410 DATA MEISTER MUELLER,DAS BRAVE LENCHEN,PLISCH UND PLUM
415 DATA DER HEILIGE ANTONIUS,FIPPS DER AFFE,MALER KLECKSEL
420 DATA KNOPP,FRAU KUEMMEL,MEISTER BOECK,HANS HUCKEBEIN
425 DATA ONKEL NOLTE
50000 CZ=1024+40*CZ
50010 POKE209,CZAND255
50020 POKE210,CZ/256
50030 POKE211,CS
50040 RETURN

READY.
```

```
P L I S C H   U N D   P L U M
- - - - - - - - - - - - - - -



BITTE GEBEN SIE IHRE BUCHSTABEN EIN


 ***** HERZLICHEN GLUECKWUNSCH! *****

 SIE HABEN DEN 'UNBEKANNTEN' BEGRIFF

         PLISCH UND PLUM

 MIT 12 VERSUCHEN ERRATEN!

 DAS SIND 80 % DER LAENGE DES
 BEGRIFFES.



 NEUES SPIEL GEFAELLIG (J/N)? N

READY.
```

L a u f s c h r i f t

Eine Folge von beliebigen Zeichen (maximal eine Stringlänge, also 255 Zeichen lang) wird als Laufschrift ausgegeben. Dazu wird mit der MID$-Funktion jeweils ein Teilstring von 40 Zeichen (entsprechend einer Bildschirmzeile) ermittelt und angezeigt. Dieser Ausschnitt aus der gesamten Zeichenkette wird bei jedem Durchlauf um ein Zeichen verschoben, sodaß eine Laufschrift entsteht, die bis zum Programmabbruch fortwährend wiederholt wird.

```
10 REM.........................................LAUFSCHRIFT
15 PRINT"{CLR}{DOWN} {RVS ON}      ERZEUGUNG EINER LAUFSCHRIFT     "
20 PRINT"{DOWN}{DOWN}{DOWN} DIE LAUFSCHRIFT SOLL LAUTEN:":INPUT A$
30 A$="                                        "+A$
40 PRINT"{CLR}": POKE 53280,0: POKE 53281,0
50 L=LEN(A$)
60 FOR X=1 TO L
70 S=L-X
80 PRINT"{HOME}{DOWN}{DOWN}{DOWN}{DOWN}{DOWN}{DOWN}{DOWN}{DOWN}{DOWN}{DOWN}{DOWN}{DOWN}";
90 PRINT"{RVS ON}"MID$(A$,X,40);
100 IF S<40 THEN PRINT MID$(A$,1,39-S)
110 FOR T=1 TO 150: NEXT T
120 NEXT X
130 GOTO 40

READY.
```

T i p p - T e s t

Was halten Sie davon, Ihre Schreibmaschinenkenntnisse mit Hilfe des Computers zu verbessern? Dieses Programm soll eine kleine Anregung für eine solche Anwendung darstellen.

Eine vorher festgelegte Anzahl von Buchstaben und sonstigen Zeichen wird der Reihe nach in zufälliger Folge angezeigt. Das jeweils sichtbare Zeichen muß möglichst schnell auf der Tastatur eingetippt werden. Hat man sich vertippt, so wartet das Programm solange, bis das richtige Zeichen eingegeben wird.

Am Ende wird die benötigte Tipp-Zeit angegeben, die zudem noch umgerechnet wird in die Zahl der Anschläge pro Minute.

```
5 REM.........................................TIPPTEST
10 PRINT"⊐▒ ▨                TIPP-TEST            "
20 PRINT"▒▒▒▒▒ WIEVIELE BUCHSTABEN SOLLEN ANGEZEIGT"
25 PRINT" WERDEN";: INPUT N: IF N=0 THEN END
30 PRINT"▒▒ O.K.!...."
35 PRINT"▒▒▮▮▮▮▮▮▮▮▮.....BITTE KONZENTRIEREN....."
40 FOR I=1 TO 2000:NEXT I
45 PRINT"⊐"
50 FOR P=1 TO N
55 REM...............................BUCHSTABEN MIT ZUFALLSZAHLEN
60 A=INT(RND(1)*26+65)
65 REM.........................................BUCHSTABENAUSGABE
70 PRINT"⊐▒▒▒▒▒▒▒▒▒▒▒▒▮▮▮▮▮▮▮▮▮▮▮▮▮";CHR$(A);CHR$(154)
90 GET B$: TI$="000000": IFB$="" THEN PC=PC+TI: GOTO 90
100 IF B$=CHR$(A) THEN TI$="000000": NEXT P: GOTO 130
110 TI$="000000": EC=EC+1: GOTO 70
120 REM.........................................AUSGABE BEWERTUNG
130 PRINT"⊐": T=PC/60
140 PRINT"▒▒";N;"BUCHSTABEN IN";T;"SEKUNDEN:"
150 PRINT:PRINT" DAS SIND";INT(N/T*60+.5);"BUCHSTABEN/MINUTE!"
160 PRINT"▒▒ FEHLER";EC;"(=";EC/N*100;"%)"
170 END

READY.
```

L e r n k a r t e n

Dieses Programm kann das 'Pauken' von Vokabeln usw. erleichtern und unterhaltsamer gestalten. Die Daten stehen paarweise in DATA-Zeilen (z.B. englisches Wort, deutsche Übersetzung). Als Endemarkierung werden zwei Sonderzeichen verwendet (*,*), sodaß es nicht notwendig ist, die Anzahl der Daten anzugeben: zu Beginn des Programms zählt der Computer selbst die Daten. Anschließend werden sie gemischt und in einer zufälligen Reihenfolge paarweise vorgelegt: das erste Wort eines Paares wird angezeigt und nach einer kurzen Pause der entsprechende zugeordnete Begriff. Nach einer weiteren Pause wird das nächste Datenpaar angezeigt, usw. bis sämtliche Daten einmal abgefragt worden sind. Anschließend kann man das Programm beenden oder aber die Liste noch einmal vorlegen lassen, nachdem die Daten wieder neu gemischt worden sind.

In der vorliegenden Programmform werden die Daten ohne Zutun des Benutzers der Reihe nach angezeigt. Die Geschwindigkeit läßt sich durch Ändern der entsprechenden Warteschleifen natürlich beliebig einstellen. Ebenso leicht kann man die Abfragen auch über die Tastatur steuern, indem man die Warteschleifen durch eine GET-Anweisung ersetzt, die bewirkt, daß erst nach einem Tastendruck das nächste Wort angezeigt wird:

```
... GET X$: IF X$ = "" THEN GOTO ...
```

```
10 REM.............................................LERN-KARTEN
20 PRINT"⌐M ⍰                LERN-KARTEN          "
30 REM............................FRAGEN UND ANTWORTEN ZAEHLEN
40 READ F$,A$
50 IF F$<>"*" THEN N=N+1: GOTO 40
60 REM.....................................ABFRAGE VORBEREITEN
70 RESTORE
80 FOR I=1 TO N
90 A(I)=I
100 NEXT I
110 REM.................................................MISCHEN
120 FOR J=1 TO N
130 X=INT((N-J+1)*RND(1)+1)
140 READ F$(A(X)),A$(A(X))
150 A(X)=A(N-J+1)
160 NEXT J
170 REM.................................................ABFRAGE
180 FOR J=1 TO N
190 PRINT: PRINT TAB(5);F$(J);
200 FOR PAUSE =1 TO 1000
210 NEXT PAUSE
220 PRINT TAB(15);"⍰ ";A$(J);" ■"
230 FOR PAUSE=1 TO 1000
240 NEXT PAUSE
250 NEXT J
260 PRINT"MMM *** ALLE";N;" WOERTER SIND EINMAL"
270 PRINT"      VORGELEGT WORDEN!"
280 PRINT: PRINT" NOCH EINMAL? (J/N)";
290 GET E$
300 IF E$="" OR (E$<>"J" AND E$<>"N") THEN GOTO 290
310 IF E$<>"N" THEN PRINT"⌐M": GOTO 70
320 END
1000 REM.........................................DATEN-PAARE
1010 DATA ONE,EINS,TWO,ZWEI,THREE,DREI,FOUR,VIER,FIVE,FUENF
1999 DATA *,*: REM..........................DATENENDE-MARKIERUNG
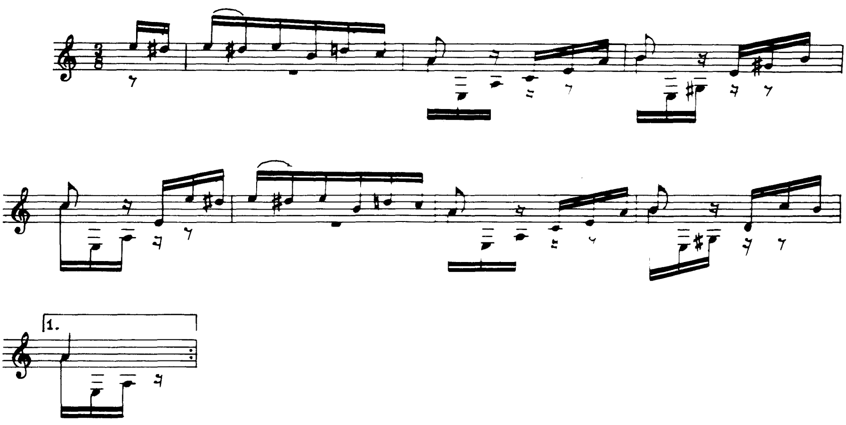
READY.
```

M u s i k - E i n g a b e - P r o g r a m m

Dieses - etwas 'länglich' geratene - Programm, hilft Ihnen bei der Eingabe von Musikstücken.

Ludwig van Beethoven (1770-1827)

„Für Elise"



Am Beispiel des ersten Taktes dieses Musikstückes wollen wir demonstrieren, wie die Eingabe der Noten erfolgt.

Zunächst wird die Notenbezeichnung zusammen mit der gewünschten Oktavlage angegeben. (Der Sound-Chip hat einen Umfang von 8 Oktaven). Darauf folgt, durch ein Komma getrennt, der Notenwert in reziproker Form, d.h. 4 für 1/4 usw.

Für die erste Note ergibt sich also: E5,16, für die zweite: D#5,16 usw. Das Vorzeichen zur Erhöhung um einen halben Ton wird also, wie in der Musik üblich, durch ein Kreuz bezeichnet. (Vorzeichen zur Erniedrigung um einen halben Ton sind in diesem Programm nicht vorgesehen: solche Töne sind durch eine Erhöhung des nächst tieferen Tones darzustellen.)

Außer der Noteneingabe haben wir noch einige weitere Möglichkeiten vorgesehen. Zunächst kann die eingegebene Melodie abgespielt werden. Ist man mit dem Resultat zufrieden, können die Notendaten abgespeichert werden. Zusätzlich könen die eingegebenen Noten auf dem Bildschirm oder auf dem Drucker ausgegeben werden. Eine Ausgabe der Noten für unser Beispiel ist am Ende des Programmes abgebildet.

Zur Änderung der eingegebenen Noten gibt es zwei Möglichkeiten:

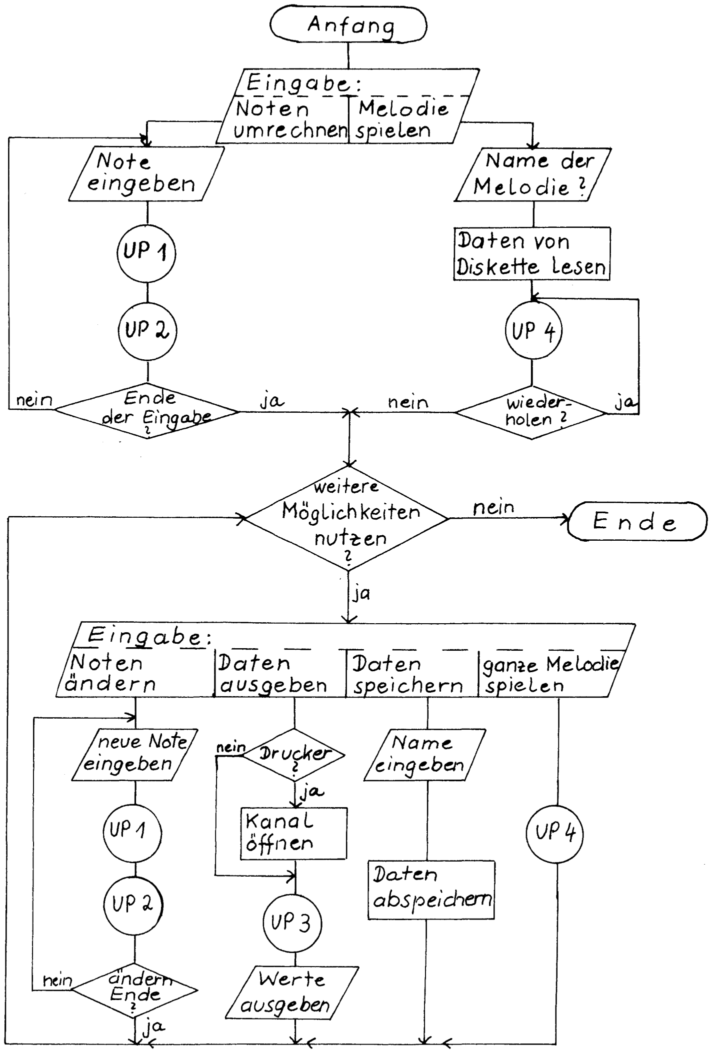
Erstens können direkt nach der Eingabe Korrekturen vorgenommen werden.

Zweitens besteht die Möglichkeit, bereits abgespeicherte Notendaten zu korrigieren, wenn sie nach dem Einlesen einmal abgespielt worden sind. (Zum nächsten Abspeichern muß das alte Datenfile gelöscht werden.)

Dazu noch einige Anmerkungen: Die Anzahl der eingegebenen Noten ist durch die entsprechende Feld-Dimensionierung begrenzt. Außerdem kann das Programm nur einstimmige Musikstücke verarbeiten. Die Anleitung zur Erzeugung mehrstimmiger Musik findet man in Kapitel 7.

Der Aufbau des Programms läßt sich im übrigen mit Hilfe des nachstehenden Flußdiagramms leicht überblicken. Weitere Erklärungen stehen im Programmlisting in Kommentarzeilen, sodaß Änderungen nicht zu schwierig sein dürften.

Die Unterprogramme stehen am Anfang des Programms, um die Ablaufgeschwindigkeit zu erhöhen. Dies ist immer dann sinnvoll, wenn Unterprogramme häufig aufgerufen werden, da der BASIC-Interpreter beim Aufruf von Unterprogrammen die Suche danach bei der niedrigsten Speicheradresse beginnt.



Unterprogramme:

UP 1	Umrechnen der eingegebenen Noten in Daten
UP 2	Spielen eines einzelnen Tones
UP 3	Umrechnen der Daten in Noten
UP 4	Spielen einer Melodie im Zusammenhang

Fluß - Diagramm: Hilfs-Programm zur Musikprogrammierung

```
10 REM.......................................MUSIK-HILFSPROGRAMM
20 DIM GF%(11),ND%(500),HF%(2000),NF%(2000),WF%(2000)
30 SI=54272
40 FOR I=SI TO SI+24: POKE I,0: NEXT I
50 POKE SI+5,30 :POKE SI+6,20
60 FOR I=0 TO 11: READ F%: GF%(I)=F%: NEXT I
65 REM ........................................BASIS-FREQUENZEN
80 DATA 268,284,301,318,337,358,379,401,425,451,477,506
90 GOTO 1000
100 REM...................................UMRECHNUNG DER EINGABE
105 IF MID$(A$,2,1)="#" THEN GOTO 150
110 IF LEFT$(A$,1)="P" THEN GOTO 175
115 IF LEFT$(A$,1)="C" THEN NT=0: GOTO 175
120 IF LEFT$(A$,1)="D" THEN NT=2: GOTO 175
125 IF LEFT$(A$,1)="E" THEN NT=4: GOTO 175
130 IF LEFT$(A$,1)="F" THEN NT=5: GOTO 175
135 IF LEFT$(A$,1)="G" THEN NT=7: GOTO 175
140 IF LEFT$(A$,1)="A" THEN NT=9: GOTO 175
145 IF LEFT$(A$,1)="H" THEN NT=11: GOTO 175
150 IF LEFT$(A$,2)="C#" THEN NT=1: GOTO 175
155 IF LEFT$(A$,2)="D#" THEN NT=3: GOTO 175
160 IF LEFT$(A$,2)="F#" THEN NT=6: GOTO 175
165 IF LEFT$(A$,2)="G#" THEN NT=8: GOTO 175
170 IF LEFT$(A$,2)="A#" THEN NT=10: GOTO 175
171 PRINT" FALSCHE EINGABE!": I=I-1: FE=1: RETURN
175 REM..............................BERECHNUNG DES DATENWERTES
180 O%=VAL(RIGHT$(A$,1))
185 B=VAL(LEFT$(B$,2))
190 NZ%=16/B
195 IF RIGHT$(B$,1)="." THEN NZ%=1.5*NZ%
200 IF LEFT$(A$,1)="P" THEN O%=0: NT=0
205 ND%(I)=(NZ%*8+O%)*16+NT
206 IF LEFT$(A$,1)="P" THEN ND%(I)=-ND%(I)
210 RETURN
```

```
250 REM..............................SPIELEN DER EINZELNEN NOTEN
260 IF ND%(I)<0 THEN NW%=-ND%(I): WA=0: WE=0: GOTO 290
265 NZ%=ND%(I)/128: O%=(ND%(I)-128*NZ%)/16
270 NW%=128*NZ%: NT=ND%(I)-128*NZ%-16*O%
275 GF=GF%(NT)*2↑O%: WA=33: WE=32
280 HF%=GF/256: NF%=GF-HF%*256
285 POKE SI+1,HF%: POKE SI,NF%
290 POKE SI+4,WA
295 FOR T=1 TO NW%/2: NEXT T
300 POKE SI+4,WE
310 RETURN
350 REM..................................UMRECHNEN DATEN -> TEXT
354 PU=0
355 IF ND%(I)<0 THEN N$="PAUSE": NZ%=-ND%(I)/128: GOTO365
360 NZ%=ND%(I)/128: O%=(ND%(I)-128*NZ%)/16
365 NT=ND%(I)-128*NZ%-16*O%: NW=16/NZ%
366 IF NW>INT(NW) THEN NW%=2*INT(NW):PU=1: GOTO 368
367 NW%=NW
368 IFN$="PAUSE"THEN RETURN
370 ON NT+1 GOTO 375,380,385,390,395,400,405,410,415,420,425,430
375 N$="C  ": RETURN
380 N$="C# ": RETURN
385 N$="D  ": RETURN
390 N$="D# ": RETURN
395 N$="E  ": RETURN
400 N$="F  ": RETURN
405 N$="F# ": RETURN
410 N$="G  ": RETURN
415 N$="G# ": RETURN
420 N$="A  ": RETURN
425 N$="A# ": RETURN
430 N$="H  ": RETURN
```

```
500 REM..............................SPIELEN EINER GANZEN MELODIE
501 REM..............................................BERECHNUNGEN
505 J=0: FOR I=0 TO N
510 WA=33: WE=32
515 ND=ND%(I): IF ND<0 THEN ND=-ND: WA=0: WE=0
520 NW%=ND/128: O%=(ND-128*NW%)/16: NT=ND-128*NW%-16*O%
525 GF=GF%(NT): GF=GF*2↑O%: HF%=GF/256: NF%=GF-256*HF%
530 IFNW%=1THENHF%(J)=HF%: NF%(J)=NF%: WF%(J)=WA: J=J+1: GOTO 555
535 FOR K=1 TO NW%-1
540 HF%(J)=HF%: NF%(J)=NF%: WF%(J)=WA: J=J+1
545 NEXT K
550 HF%(J)=HF%: NF%(J)=NF%: WF%(J)=WE
555 J=J+1
560 NEXT I: JMAX=J
565 PRINT" ENDE DER BERECHNUNGEN! AUF TASTENDRUCK"
570 PRINT" KOENNEN SIE DIE MELODIE JETZT HOEREN."
575 GET HM$: IF HM$="" THEN GOTO 575
580 REM...................................................SPIELEN
585 POKE SI+24,15
590 FOR J=0 TO JMAX
595 POKE SI,NF%(J): POKE SI+1,HF%(J): POKE SI+4,WF%(J)
600 FOR T=1 TO 40: NEXT T
605 NEXT J
610 FOR T=1 TO 50: NEXT T
615 POKE SI+24,0
620 RETURN
1000 REM.........................................HAUPTPROGRAMM
1010 PRINT"▒▒ ▒         MUSIK-HILFSPROGRAMM          "
1020 PRINT"▒▒▒ MIT DIESEM PROGRAMM KOENNEN SIE NOTEN"
1025 PRINT" IN DATA-WERTE UMRECHNEN ▒ U ▒"
1030 PRINT"▒ ODER"
1035 PRINT"▒ BEREITS ABGESPEICHERTE MELODIEN VON"
```

```
1040 PRINT" DER DISKETTE EINLESEN UND SPIELEN ▒ S ■"
1045 PRINT"⊠ LASSEN."
1050 PRINT"⊠⊠⊠ BITTE DRUECKEN SIE DIE TASTE (U/S),"
1055 PRINT" DIE IHRER WAHL ENTSPRICHT."
1060 GET Z$: IF Z$="" THEN GOTO 1060
1065 IF Z$<>"U" AND Z$<>"S" THEN GOTO 1060
1070 IF Z$="U" THEN GOTO 1300
1100 REM....................SPIELEN EINER ABGESPEICHERTEN MELODIE
1105 PRINT"⌂⊠⊠⊠⊠ UM EINE BEREITS ABGESPEICHERTE MELODIE"
1110 PRINT"⊠ SPIELEN ZU KOENNEN, LEGEN SIE BITTE"
1115 PRINT"⊠ DIE RICHTIGE DISKETTE EIN UND GEBEN"
1120 PRINT"⊠ ANSCHLIESSEND DEN NAMEN DER MELODIE AN."
1125 PRINT: PRINT: INPUT" NAME";XY$
1130 OPEN 13,8,2,XY$+",S,R"
1135 INPUT#13,N
1140 FOR I=0 TO N
1145 INPUT#13,ND%(I)
1150 NEXT I
1155 CLOSE 13
1160 PRINT" GEDULD, ERST MUSS GERECHNET WERDEN."
1165 GOSUB 500
1170 PRINT"⊠⊠ WIEDERHOLUNG (J/N)?"
1175 GET Z$: IF Z$="" THEN GOTO 1175
1180 IF Z$<>"J" AND Z$<>"N" THEN GOTO 1175
1185 IF Z$="J" THEN GOSUB 580
1190 GOTO 1900
1300 REM....................................NOTENEINGABE-PROGRAMM
1305 PRINT"⌂⊠ GEBEN SIE DIE NOTEN BITTE IN FOLGENDER"
1310 PRINT" SCHREIBWEISE EIN:"
1315 PRINT"⊠ NOTE EV.# OKT., REZIPROKER NOTENWERT"
1320 PRINT"⊠ ALSO Z.B. ▒ G5,4 ■"
1325 PRINT" (G IN DER 5. OKTAVE ALS VIERTELNOTE)"
```

```
1330 PRINT" ODER ▣ F#3,2. ■"
1335 PRINT" (FIS IN DER 3. OKTAVE ALS PUNKTIERTE"
1340 PRINT" HALBE NOTE).
1345 PRINT"▩ PAUSEN KOENNEN EINGEGEBEN WERDEN DURCH"
1350 PRINT" P,PAUSENLAENGE, ALSO ETWA ▣ P,8 ■ ALS"
1355 PRINT" 1/8-PAUSE."
1360 PRINT"▩ WOLLEN SIE DIE EINGABE BEENDEN, GEBEN"
1365 PRINT" SIE ANSTELLE EINER NOTE ▣ 0,0 ■ EIN."
1370 PRINT"▩ DANACH STEHEN IHNEN DANN VERSCHIEDENE"
1375 PRINT" MOEGLICHKEITEN DER WEITERARBEIT ZUR"
1380 PRINT" VERFUEGUNG."
1385 PRINT"▩ WEITER MIT TASTENDRUCK."
1390 GET Z$: IF Z$="" THEN GOTO 1390
1400 REM............................................NOTENEINGABE
1405 PRINT"▢▩ BEGINNEN SIE NUN MIT DER EINGABE."
1410 I=0
1415 PRINT"▩ NOTE NR.";I
1420 INPUT"NOTENWERT"; A$,B$: IF A$="0" THEN N=I-1: GOTO 1450
1421 WB=VAL(B$)
1422 IF NOT(WB<>1ANDWB<>2ANDWB<>4ANDWB<>8ANDWB<>16) THEN GOTO1425
1423 PRINT"FALSCHE EINGABE!": GOTO 1415
1425 GOSUB 100
1426 IF FE=1 THEN FE=0: GOTO 1445
1430 POKE SI+24,15
1435 GOSUB 250
1440 POKE SI+24,0
1445 IF I<500 THEN I=I+1: GOTO 1415
1450 PRINT"▢▩▩ NACHDEM SIE NUN IHRE MELODIE EINGEGEBEN"
1455 PRINT"▢ HABEN, STEHEN IHNEN FOLGENDE MOEGLICH-"
1460 PRINT" KEITEN ZUR VERFUEGUNG:"
1465 PRINT"▩ AENDERN EINZELNER NOTEN    ▣ A ■"
1470 PRINT" AUSGABE DER DATEN-WERTE    ▣ D ■"
```

```
1475 PRINT" ABSPEICHERN AUF DISKETTE  ▣ S ■"
1480 PRINT" HOEREN DER GANZEN MELODIE ▣ M ■"
1485 PRINT"⊠⊠ BITTE DRUECKEN SIE DIE ENTSPRECHENDE"
1490 PRINT" TASTE."
1495 GET Z$: IF Z$="" THEN GOTO 1495
1500 IF Z$<>"A"ANDZ$<>"D"ANDZ$<>"S"ANDZ$<>"M" THEN GOTO 1495
1505 IF Z$="D" THEN GOTO 1600
1510 IF Z$="S" THEN GOTO 1700
1515 IF Z$="M" THEN GOTO 1800
1520 REM................................AENDERN EINZELNER NOTEN
1525 PRINT"⊐⊠⊠ ZUM AENDERN EINZELNER NOTEN GEBEN SIE"
1530 PRINT" DIE NUMMER DER NOTE AN, DIE GEAENDERT"
1535 PRINT" WERDEN SOLL, UND DANACH IN DER GEWOHN-"
1540 PRINT" TEN SCHREIBWEISE DIE NOTE."
1545 INPUT"⊠⊠ NOTENNR.";I
1550 INPUT" NOTE";A$,B$
1555 GOSUB 100
1560 POKE SI+24,15
1565 GOSUB 250
1570 POKE SI+24,0
1575 PRINT" SOLL NOCH ETWAS GEAENDERT WERDEN (J/N)?"
1580 GET Z$: IF Z$="" THEN GOTO 1580
1585 IF Z$="J"THEN GOTO 1545
1590 IF Z$<>"J" THEN GOTO 1900
1600 REM................................AUSGABE DER DATEN-WERTE
1605 PRINT"⊐⊠ SOLLEN DIE DATEN AUF DEM DRUCKER ▣ D ■"
1610 PRINT"⊠ ODER AUF DEM BILDSCHIRM ▣ B ■ AUSGEGEBEN"
1615 PRINT"⊠ WERDEN?"
1620 PRINT"⊠⊠ BITTE DRUECKEN SIE DIE ENTSPRECHENDE"
1625 PRINT" TASTE."
1630 GET Z$: IF Z$="" THEN GOTO 1630
1635 IF Z$<>"D" AND Z$<>"B" THEN GOTO 1630
```

```
1640 PRINT"{CLR}{DOWN}":IF Z$="D" THEN OPEN 1,4: CMD1
1645 PRINT"NR.";TAB(4);"NOTE";TAB(10);"OKT.";TAB(16);"WERT";
1650 PRINT TAB(23);"DATE"
1655 FOR I=0 TO N: GOSUB 350
1660 PRINTI;TAB(5);N$;TAB(10);
1665 IF N$="PAUSE" THEN GOTO 1675
1670 PRINT O%;
1675 PRINT TAB(16);"1/";STR$(NW%);
1676 IF PU<>0 THEN PRINT".";
1677 PRINT TAB(22);ND%(I)
1680 IF Z$="D" THEN GOTO 1690
1684 IF I=0 OR (INT(I/20)-I/20)<>0 THEN GOTO 1690
1685 PRINT" WEITERE DATEN (J/N)?"
1686 GET A$: IF A$="" THEN GOTO 1686
1687 IF A$="N" THEN GOTO 1900
1690 NEXT I: IF Z$="D" THEN PRINT#1:CLOSE 1
1695 PRINT" DRUECKEN SIE EINE TASTE, WENN DAS"
1696 PRINT" PROGRAMM WEITERGEHEN KANN."
1697 GET A$: IF A$="" THEN GOTO 1697
1698 GOTO 1900
1700 REM...............................ABSPEICHERN DER DATENWERTE
1705 PRINT"{CLR}{DOWN} BEVOR DIE DATEN ABGESPEICHERT WERDEN"
1710 PRINT"{DOWN} KOENNEN, MUESSEN SIE EINE DISKETTE"
1715 PRINT"{DOWN} EINLEGEN UND DEN NAMEN IHRER MELODIE"
1720 PRINT"{DOWN} ANGEBEN."
1725 INPUT"{DOWN}{DOWN}{DOWN} NAME";XY$
1730 OPEN 13,8,2,XY$+",S,W"
1735 PRINT#13,N
1740 FOR I=0 TO N
1745 PRINT#13,ND%(I)
1750 NEXT I
1755 CLOSE 13
1760 GOTO 1900
```

```
1800 REM.............................SPIELEN DER GANZEN MELODIE
1805 PRINT" HABEN SIE GERADE AN DEN NOTEN ETWAS"
1810 PRINT" GEAENDERT BZW. MOECHTEN SIE IHRE"
1811 PRINT" MELODIE ZUM ERSTEN MAL HOEREN? (J/N)"
1815 GET NG$: IF NG$="" THEN GOTO 1815
1820 IF NG$<>"J" AND NG$<>"N" THEN GOTO 1815
1825 IF NG$="N" THEN GOSUB 580: GOTO 1900
1830 PRINT" DANN MUESSEN SIE EINEN AUGENBLICK"
1835 PRINT" GEDULD HABEN, ICH MUSS ERST RECHNEN."
1840 GOSUB 500
1900 REM......................................ABFRAGE OB WEITER
1905 PRINT" WOLLEN SIE NOCH ANDERE MOEGLICH-"
1910 PRINT" KEITEN AUSNUTZEN (J/N)?"
1915 GET C$: IF C$="" THEN GOTO 1915
1920 IF C$<>"J"ANDC$<>"N" THEN GOTO 1915
1925 IF C$="J" THEN PRINT"": GOTO 1465
1930 IF C$="N" THEN END

READY.
```

NR.	NOTE	OKT.	WERT	DATE
0	E	5	1/ 16	212
1	D#	5	1/ 16	211
2	E	5	1/ 16	212
3	D#	5	1/ 16	211
4	E	5	1/ 16	212
5	H	4	1/ 16	203
6	D	5	1/ 16	210
7	C	5	1/ 16	208
8	A	4	1/ 16	201
9	E	3	1/ 16	180
10	A	3	1/ 16	185
11	C	4	1/ 16	192
12	E	4	1/ 16	196
13	A	4	1/ 16	201
14	H	4	1/ 16	203
15	E	3	1/ 16	180
16	G#	3	1/ 16	184
17	E	4	1/ 16	196
18	G#	4	1/ 16	200
19	H	4	1/ 16	203
20	C	5	1/ 16	208
21	E	3	1/ 16	180
22	A	3	1/ 16	185
23	E	4	1/ 16	196
24	E	5	1/ 16	212
25	D#	5	1/ 16	211
26	E	5	1/ 16	212
27	D#	5	1/ 16	211
28	E	5	1/ 16	212
29	H	4	1/ 16	203
30	D	5	1/ 16	210
31	C	5	1/ 16	208
32	A	4	1/ 16	201
33	E	3	1/ 16	180
34	A	3	1/ 16	185
35	C	4	1/ 16	192
36	E	4	1/ 16	196
37	A	4	1/ 16	201
38	H	4	1/ 16	203
39	E	3	1/ 16	180
40	G#	3	1/ 16	184
41	D	4	1/ 16	194
42	C	5	1/ 16	208
43	H	4	1/ 16	203
44	A	4	1/ 4	585

Anhang

Anhang

Speicherbelegung des Commodore 64

Die Tabelle gibt einen Überblick über die verschiedenenen Speicherbereiche des Commodore 64.

In den Kapiteln 5 und 9 wurde einiges über die Aufteilung der ROM- und RAM-Speicher gesagt. Insbesondere die Doppelbelegung von Speicheradressen läßt ja einige interessante Anwendungen zu. Im folgenden wird die Speicherbelegung noch einmal zusammenfassend beschrieben.

0 - 827 Diese Speicher sind für wichtige Aufgaben des Betriebssystems zuständig (Anwendung s. Kapitel 8).

828 - 1019 Hier befindet sich der Kassettenpuffer. Alle Daten, die von der Kassette übernommen werden, werden hier zwischengespeichert, bevor sie z.B. in Programmen weiterverarbeitet werden. Benutzt man ein Diskettenlaufwerk, können in diese Speicherplätze Daten oder kurze Maschinensprache-Programme abgelegt werden.

1024 - 2023 Diese Speicherplätze belegt der Bildschirmspeicher (s. Kapitel 5 und 8).

2040 - 2047 Bei der Benutzung von Sprites sind diese Speicherplätze notwendig. Sie zeigen jeweils auf den Datenblock für ein Sprite.

2048 - 40959 In diesem Bereich stehen die BASIC-Programme (s. auch Kapitel 5 und 9).

40960 - 49151 Dieser ROM-Bereich von 8 K enthält den BASIC-Interpreter.

49152 - 53247 Hier befindet sich ein weiterer RAM-Bereich, der von BASIC jedoch nicht benutzt wird. In diesen Speicherplätzen werden umfangreichere Maschinenprogramme abgelegt. Auch bis zu 4 K Daten können hier untergebracht werden.

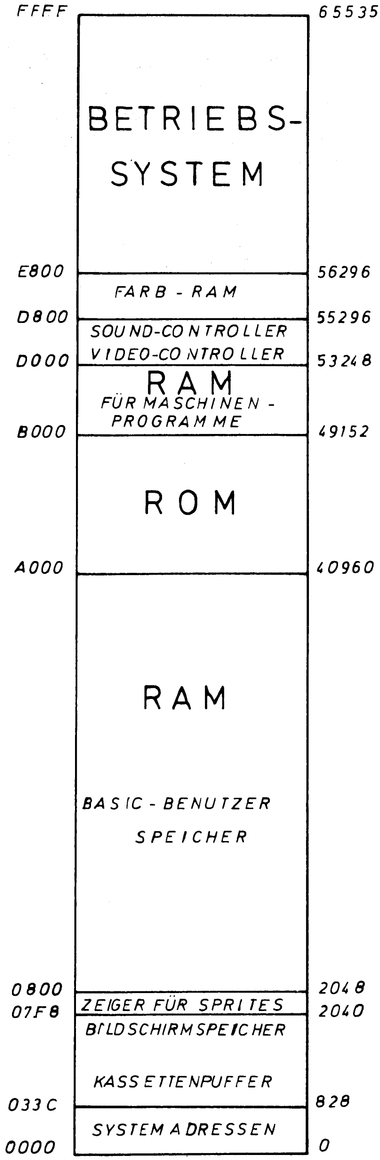
53248 - 55296 Diese Speicherplätze kontrollieren u.a. den Video- und den Sound-Chip. (Näheres dazu in den Kapiteln 5 bis 9).

55296 - 56295 Dies ist der Farbspeicher-Bereich. Im Gegensatz zum Bildschirmspeicher läßt er sich nicht verschieben (s. Kapitel 5).

56296 - 65535 In diesem Bereich befinden sich Systemroutinen des Computers. Diese steuern z.B. die Ein- und Ausgabe, die Systemuhr, den Tastaturpuffer. Diese Systemroutinen lassen sich sich auch in eigenen Maschinenprogrammen verwenden.

Die Aufteilung und Funktion einzelner Speicherbereiche ist in den folgenden Tabellen dargestellt.

SPEICHERBELEGUNG DES COMMODORE 64



D e r   B i l d s c h i r m s p e i c h e r

Die Tabelle zeigt die Verteilung der Speicheradressen von 1024 bis 2023, also der Speichergruppe, die für die Darstellung von Zeichen an den entsprechenden Bildschirmplätzen sorgt. Um eine bestimmte Bildschirmstelle zu finden, muß man zu den angegebenen Werten 1000 addieren und bis zur gewünschten Stelle weiterzählen.

Beispiel:

```
10 FOR I = 1 TO 40
20 POKE 1024+I,81
30 NEXT I
```

Durch diese Anweisungen wird in der obersten Bildschirmzeile eine Reihe von Kreisen erzeugt, die allerdings bisher noch unsichtbar sind, da sie zunächst in der Hintergrundfarbe dargestellt werden. Um sie sichtbar zu machen, muß noch eine andere Farbe (als die Hintergrundfarbe) in die entsprechenden Farbspeicher gebracht werden (s. Farbspeicher-Tabelle).

BILDSCHIRMSPEICHER

		10	20	30	40
1000+	24	33	43	53	63
	64	73	83	93	103
	104	113	123	133	143
	114	153	163	173	183
5	184	193	203	213	223
	224	233	243	253	263
	264	273	283	293	303
	304	313	323	333	343
	344	353	363	373	383
10	384	393	403	413	423
	424	433	443	453	463
	474	473	483	493	503
	504	513	523	533	543
	544	553	563	573	583
15	584	593	603	613	623
	624	633	643	653	663
	664	673	683	693	703
	704	713	723	733	743
	744	753	763	773	783
20	784	793	803	813	823
	824	833	843	853	863
	864	873	883	893	903
	904	913	923	933	943
	944	953	963	973	983
25	984	993	1003	1013	1023

D e r   F a r b s p e i c h e r

Der Farbspeicher befindet sich in den Adressen von 55296 bis 56295 (= 1000 Speicherplätze, d.h. ein Speicherplatz für jede der 40 x 25 Bildschirmpositionen). Aus diesen Speicherplätzen entnimmt der Computer die Information, welche Farbe ein (beliebiges) Zeichen an einer bestimmten Stelle des Bildschirms haben soll.

Ein Zeichen kann auf dem Bildschirm nur dann erscheinen, wenn außer der Bildschirmposition auch eine Farbe für die betreffende Stelle angegeben wird. Ohne eine Farbangabe werden die Zeichen in der Hintergrundfarbe dargestellt - sind also nicht zu sehen!

Das Beispiel in der Beschreibung der Bildschirmspeicher-Tabelle muß also um eine Farbangabe erweitert werden, z.B. durch:

```
25 POKE 55296+I,7
```

Jetzt wird die Reihe aus Kreisen in gelber Farbe sichtbar.

Die Code-Werte für die Farben finden Sie in der Übersicht hinter der Farbspeicher-Tabelle.

FARBSPEICHER

										10										20										30										40
55200+	96									105										115										125										135
	136									145										155										165										175
	176									185										195										205										215
	216									225										235										245										255
5	256									265										275										285										295
	296									305										315										325										335
	336									345										355										365										375
	376									385										395										405										415
	416									425										435										445										455
10	456									465										475										485										495
	496									505										515										525										535
	536									545										555										565										575
	576									585										595										605										615
	616									625										635										645										655
15	656									665										675										685										695
	696									705										715										725										735
	736									745										755										765										775
	776									785										795										805										815
	816									825										835										845										855
20	856									865										875										885										895
	896									905										915										925										935
	936									945										955										965										975
	976									985										995										1005										1015
	1016									1025										1035										1045										1055
25	1056									1065										1075										1085										1095

FARBCODES

0	SCHWARZ
1	WEIß
2	ROT
3	TÜRKIS
4	VIOLETT
5	GRÜN
6	BLAU
7	GELB
8	ORANGE
9	BRAUN
10	HELLROT
11	GRAU 1
12	GRAU 2
13	HELLGRÜN
14	HELLBLAU
15	GRAU 3

Der Grafik-Speicher

Für die Darstellung in der hochauflösenden Grafik ist der Grafik-Speicher (8192 bis 16191) zuständig. Durch die Werte dieser Speicherplätze wird bestimmt, an welchen Bildschirmstellen Punkte gesetzt werden.

Die erste Speicheradresse (8192) bezieht sich auf die erste Reihe des normalen Bildschirmkästchens in der linken oberen Ecke, die achte Adresse bezieht sich auf die letzte Reihe dieses Kästchens, die neunte auf die erste Reihe des rechts daneben liegenden, usw. Diese Art der Speicheraufteilung weicht deutlich von der von den Sprites bekannten ab!

Um Ihnen das Auffinden der Speicherplätze zu erleichtern, z.B. wenn Sie einzelne Zeichen definieren wollen, ist die Speichereinteilung in der folgenden Tabelle dargestellt.

Die Aufgabe eines Farbspeichers übernimmt in der hochauflösenden Darstellung der Bildschirmspeicher. (Deshalb erscheinen die Systemmeldungen im Grafik-Modus in Farbe).

GRAFIKSPEICHER

										10										20										30										40
8192	192	200					240					280					320					360					400					440					480			504
8512	512	520					560					600					640					680					720					760					800			824
8832	832	840					880					920					960					1000					1040					1080					1120			1144
9152	152	160					200					240					280					320					360					400					440			464
9472	472	480					520					560					600					640					680					720					760			884
9792	792	800					840					880					920					960					1000					1040					1080			1104
10112	112	120					160					200					240					280					320					360					400			424
10432	432	440					480					520					560					600					640					680					720			744
10752	752	760					800					840					880					920					960					1000					1040			1064
11072	072	080					120					160					200					240					280					320					360			384
11392	392	400					440					480					520					560					600					640					680			704
11712	712	720					760					800					840					880					920					960					1000			1024
12032	032	040					080					120					160					200					240					280					320			344
12352	352	360					400					440					480					520					560					600					640			664
12672	672	680					720					760					800					840					880					920					960			984
12992	992	1000					1040					1080					1120					1160					1200					1240					1280			1304
13312	312	320					360					400					440					480					520					560					600			624
13632	632	640					680					720					760					800					840					880					920			944
13952	952	960					1000					1040					1080					1120					1160					1200					1240			1264
14272	272	280					320					360					400					440					480					520					560			584
14592	592	600					640					680					720					760					800					840					880			904
14912	912	920					960					1000					1040					1080					1120					1160					1200			1224
15232	232	240					280					320					360					400					440					480					520			544
15552	552	560					600					640					680					720					760					800					840			864
15872	872	880					920					960					1000					1040					1080					1120					1160			1184

Sprite - Datenblatt

Das Sprite-Datenblatt besteht aus einem Raster von 24x21 Kästchen, die den Punkten entsprechen, aus denen sich ein Sprite zusammensetzt.

Hat man eine Figur in das Raster eingezeichnet, werden die Daten wie folgt berechnet: Die Zahlen am oberen Rand der Tabelle, die über ausgefüllten Kästchen einer Gruppe stehen, werden addiert und in die Datentabelle übertragen. (Näheres dazu in Kapitel 6 und 9).

Ist das Datenblatt vollständig ausgefüllt, kann man ein Sprite - wie in Kapitel 6 beschrieben - auf dem Bildschirm erscheinen lassen.

Eine Folie mit dem oben besprochenen Raster sowie ein Hilfsprogramm zur Erstellung von Sprites auf dem Bildschirm findet sich in unserem Grafikbuch zu diesem Rechner.

SPRITE - DATENBLATT

	TEIL1								TEIL 2								TEIL 3							
	128	64	32	16	8	4	2	1	128	64	32	16	8	4	2	1	128	64	32	16	8	4	2	1
1																								
2																								
3																								
4																								
5																								
6																								
7																								
8																								
9																								
10																								
11																								
12																								
13																								
14																								
15																								
16																								
17																								
18																								
19																								
20																								
21																								
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24

DATEN

SP. 1	SP. 2	SP. 3

Speicherübersicht: Sprites

Die nächsten Tabellen enthalten alle für die Programmierung von Sprites wichtigen Speicherplätze und deren Funktionen. Einige Anwendungen dazu befinden sich in Kapitel 6.

SPEICHERTABELLEN - SPRITES
(V=53248)

V+

0 1	SPRITE	0	X -KOORDINATE Y -KOORDINATE	
2 3	SPRITE	1	X - Y -	" "
4 5	SPRITE	2	X - Y -	"
6 7	SPRITE	3	X - Y -	"
8 9	SPRITE	4	X - Y -	"
10 11	SPRITE	5	X - Y -	"
12 13	SPRITE	6	X - Y -	"
14 15	SPRITE	7	X - Y -	"

V+

39	SPRITE	0	FARBE
40	SPRITE	1	"
41	SPRITE	2	"
42	SPRITE	3	"
43	SPRITE	4	"
44	SPRITE	5	"
45	SPRITE	6	"
46	SPRITE	7	"

V+16

	SPRITE: X-KOORDINATE über 255							
	128	64	32	16	8	4	2	1
NR.:	7	6	5	4	3	2	1	0

V+21

	SPRITE: EIN-UND AUSSCHALTEN							
	128	64	32	16	8	4	2	1
NR.:	7	6	5	4	3	2	1	0

V+23

	SPRITE: IN Y-VERGRÖSSERN							
	128	64	32	16	8	4	2	1
NR.:	7	6	5	4	3	2	1	0

V+27

	SPRITE: HINTERGRUND-PRIORITÄT							
	128	64	32	16	8	4	2	1
NR.:	7	6	5	4	3	2	1	0

V+28

	SPRITE: MEHRFARBIG							
	128	64	32	16	8	4	2	1
NR.:	7	6	5	4	3	2	1	0

V+29

	SPRITE: IN X-VERGRÖSSERN							
	128	64	32	16	8	4	2	1
NR.:	7	6	5	4	3	2	1	0

V+30

	SPRITE: MIT SPRITE KOLLISION							
	128	64	32	16	8	4	2	1
	7	6	5	4	3	2	1	0

V+31

	SPRITE:HINTERGRUND KOLLISION							
	128	64	32	16	8	4	2	1
NR.:	7	6	5	4	3	2	1	0
V+37								
	SPRITE:MEHRFARBEN NR.: 0							
	FARBCODES							
	FÜR ALLE SPRITES							
V+38								
	SPRITE:MEHRFARBEN NR.: 1							
	FARBCODES							
	FÜRALLE SPRITES							

	2047	2046	2045	2044	2043	2042	2041	2040
NR.:	7	6	5	4	3	2	1	0

Speicherübersicht: Musik

Die für die Programmierung von Musik wichtigen Speicher sind in den folgenden Tabellen angegeben. Die Aufgaben der einzelnen Adressen sind umfassend in Kapitel 7 besprochen. In Kapitel 10 finden Sie ein Hilfsprogramm, das die Eingabe von einfachen Melodien erheblich vereinfacht.

SPEICHERTABELLEN-MUSIK
(SI=54272)

1. STIMME

SI	FREQUENZ LO-BYTE	0 bis 255
SI+1	dito HI-BYTE	"
SI+2	PULSBREITE LO-BYTE	gilt nur bei RECHTECK 0 bis 255
SI+3	dito HI-BYTE	dito 0 bis 15
SI+4	WELLENFORM	RAUSCHEN RECHTECK SÄGEZAHN DREIECK 129 65 33 17
SI+5	ANSCHLAG ABSCHWELLEN	16*AN+AB; AN,AB≤15
SI+6	AUSHALTEN ABFALLEN	16*AS+AF, AS,AF≤15

2. STIMME

SI+7	FREQUENZ LO-BYTE	0 bis 255
SI+8	dito HI-BYTE	"
SI+9	PULSBREITE LO-BYTE	gilt nur bei RECHTECK 0 bis 255
SI+10	dito HI-BYTE	dito 0 bis 15
SI+11	WELLENFORM	RAUSCHEN RECHTECK SÄGEZAHN DREIECK 129 65 33 17
SI+12	ANSCHLAG ABSCHWELLEN	16*AN+AB; AN,AB≤15
SI+13	AUSHALTEN ABFALLEN	16*AS+AF, AS,AF≤15

3. STIMME

SI+14	FREQUENZ LO-BYTE	0 bis 255
SI+15	dito HI-BYTE	"
SI+16	PULSBREITE LO-BYTE	gilt nur bei RECHTECK 0 bis 255
SI+17	dito HI BYTE	dito 0 bis 15
SI+18	WELLENFORM	RAUSCHEN RECHTECK SÄGEZAHN DREIECK 129 65 33 17
SI+19	ANSCHLAG ABSCHWELLEN	16*AN+AB; AN,AB≤15
SI+20	AUSHALTEN ABFALLEN	16*AS+AF; AS,AF≤15

SI + 24

STIMME1 bis STIMME3 1-15 (max. Lautstärke)				TIEFPAß	BANDPAß	HOCHPAß	STIMME 3 aus
1	2	4	8	16	32	64	128

SI + 23

STIMME1	STIMME2	STIMME3	Extern	Resonanz			
1	2	4	8	16	32	64	128

SI + 21	SI + 22
Filterfrequenz Low-Nybble	Filterfrequenz High-Byte

Frequenz - Tabelle

Die folgende Tabelle enthält eine vollständige Liste der Notennummern, der Notenbezeichnungen mit Angabe der Oktave sowie die beiden Werte für das High- und Low-Byte zum Erzeugen des jeweiligen Tones.

Noten-nummer	Note und Oktave	Frequenzwert Dezimal	HI	LOW
0	C-0	268	1	12
1	C#-0	284	1	28
2	D-0	301	1	45
3	D#-0	318	1	62
4	E-0	337	1	81
5	F-0	358	1	102
6	F-#0	379	1	123
7	G-0	401	1	145
8	G-#0	425	1	169
9	A-0	451	1	195
10	A#-0	477	1	221
11	H-0	506	1	250
16	C-1	536	2	24
17	C#-1	568	2	56
18	D-1	602	2	90
19	D#-1	637	2	125
20	E-1	675	2	163
21	F-1	716	2	204
22	F#-1	758	2	246
23	G-1	803	3	35
24	G#-1	851	3	83
25	A-1	902	3	134
26	A#-1	955	3	187
27	H-1	1012	3	244
32	C-2	1072	4	48
33	C#-2	1136	4	112
34	D-2	1204	4	180
35	D#-2	1275	4	251
36	E-2	1351	5	71
37	F-2	1432	5	152
38	F#-2	1517	5	237
39	G-2	1607	6	71
40	G#-2	1703	6	167
41	A-2	1804	7	12
42	A#-2	1911	7	119
43	H-2	2025	7	233
48	C-3	2145	8	97
49	C#-3	2273	8	225
50	D-3	2408	9	104
51	D#-3	2551	9	247
52	E-3	2703	10	143
53	F-3	2864	11	48
54	F#-3	3034	11	218
55	G-3	3215	12	143
56	G#-3	3406	13	78

Noten-nummer	Note und Oktave	Frequenzwert Dezimal	HI	LOW
57	A-3	3608	14	24
58	A♯-3	3823	14	239
59	H-3	4050	15	210
64	C-4	4291	16	195
65	C♯-4	4547	17	195
66	D-4	4817	18	209
67	D♯-4	5103	19	239
68	E-4	5407	21	31
69	F-4	5728	22	96
70	F♯-4	6069	23	181
71	G-4	6430	25	30
72	G♯-4	6812	26	156
73	A-4	7217	28	49
74	A♯-4	7647	29	223
75	H-4	8101	31	165
80	C-5	8583	33	135
81	C♯-5	9094	35	134
82	D-5	9634	37	162
83	D♯-5	10207	39	223
84	E-5	10814	42	62
85	F-5	11457	44	193
86	F♯-5	12139	47	107
87	G-5	12860	50	60
88	G♯-5	13625	53	57
89	A-5	14435	56	99
90	A♯-5	15294	59	190
91	H-5	16203	63	75
96	C-6	17167	67	15
97	C♯-6	18188	71	12
98	D-6	19269	75	69
99	D♯-6	20415	79	191
100	E-6	21629	84	125
101	F-6	22915	89	131
102	F♯-6	24278	94	214
103	G-6	25721	100	121
104	G♯-6	27251	106	115
105	A-6	28871	112	199
106	A♯-6	30588	119	124
107	H-6	32407	126	151
112	C-7	34334	134	30
113	C♯-7	36376	142	24
114	D-7	38539	150	139
115	D♯-7	40830	159	126
116	E-7	43258	168	250
117	F-7	45830	179	6

Noten- nummer	Note und Oktave	Frequenzwert Dezimal	 HI	 LOW
118	F♯-7	48556	189	172
119	G-7	51443	200	243
120	G♯-7	54502	212	230
121	A-7	57743	225	143
122	A♯-7	61176	238	248
123	H-7	64814	253	46

Die Commodore - Zeichen

Die folgenden Seiten geben eine Übersicht über die Commodore-Grafik-Zeichen.

In der Tabelle ist angegeben, auf welcher Buchstaben-Taste das jeweilige Zeichen zu finden ist, sowie der CHR$-Wert des Zeichens und die POKE-Werte des Zeichens in der normalen und inversen Darstellung. Hierbei wurden einige im Commodore-Handbuch vorkommende Abweichungen zwischen CHR$-Wert und dazugehörigem Zeichen richtiggestellt.

POKE und CHR $ - WERTE

ZEICHEN	TASTE	POKEWERT normal	invers	CHR $
		64	192	96
	A	65	193	97
	B	66	194	98
	C	67	195	99
	D	68	196	100
	E	69	197	101
	F	70	198	102
	G	71	199	103
	H	72	200	104
	I	73	201	105
	J	74	202	106
	K	75	203	107
	L	76	204	108
	M	77	205	109
	N	78	206	110
	O	79	207	111
	P	80	208	112
	Q	81	209	113
	R	82	210	114
	S	83	211	115
	T	84	212	116

ZEICHEN	TASTE	POKEWERT normal	POKEWERT invers	CHR $
	U	85	213	117
	V	86	214	118
	W	87	215	119
	X	88	216	120
	Y	89	217	121
	Z	90	218	122
		91	219	123
		92	220	124
		93	221	125
 		94	222	126
 		95	223	127
LEERZEICHEN		96	224	
		97	225	161
		98	226	162
		99	227	163
		100	228	164
		101	229	165
		102	230	166
		103	231	167
		104	232	168
 		105	233	169

ZEICHEN	TASTE	POKEWERT normal	POKEWERT invers	CHR $
		106	234	170
		107	235	171
		108	236	172
		109	237	173
		110	238	174
		111	239	175
		112	240	176
		113	241	177
		114	242	178
		115	243	179
		116	244	180
		117	245	181
		118	246	182
		119	247	183
		120	248	184
		121	249	185
		122	250	186
		123	251	187
		124	252	188
		125	253	189
		126	254	190
		127	255	191

F e h l e r m e l d u n g e n

Wenn Sie sich schon etwas mit Ihrem Computer beschäftigt haben, werden Ihnen viele der folgenden Begriffe mit Sicherheit bekannt vorkommen. Hier geht es nämlich um die Fehlermeldungen des BASIC-Interpreters. Die 'ach so ärgerlichen' Fehlermeldungen ermöglichen es Ihnen, viele Programmierfehler zu erkennen und zu korrigieren. Die Fehlermeldungen des BASIC-Interpreters werden in alphabetischer Reihenfolge aufgeführt.

BAD DATA

Es wurde versucht, von einem File String-Werte einzulesen. In dem Programm wurden jedoch numerische Daten erwartet, z.B.:

```
READ A,W,B
```

obwohl in dem Datenfile A$,W$,B$ abgespeichert wurde.

BAD SUBSCRIPT

Sie verwenden eine indizierte Variable, deren Index größer ist, als in der DIM-Anweisung vorgesehen wurde.

CAN'T CONTINUE

Der Befehl CONT kann nicht ausgeführt werden, wenn ein Programm unterbrochen und anschließend eine Zeile verändert wurde. Diese Fehlermeldung erscheint ebenfalls, wenn - nach CONT - ein Fehler auftrat oder das Programm nicht mit RUN gestartet wurde.

DEVICE NOT PRESENT

Es wurde versucht, ein Peripheriegerät (z.B. Disk) anzusprechen, das entweder nicht eingeschaltet oder nicht vorhanden ist, z.B.:

```
SAVE "TEST,9
```

Wenn kein zweites Laufwerk (mit der Gerätenummer 9) vorhanden ist, erfolgt die Fehlermeldung.

DIVISION BY ZERO

Eine Division durch Null ist mathematisch nicht definiert und daher unzulässig.

EXTRA IGNORED

In einer INPUT-Anweisung wurden zuviele Variablen eingegeben. Der Rechner ignoriert die überzähligen Variablen und setzt das Programm fort.

FILE NOT FOUND

Es wurde versucht, ein File mit einem bestimmten Namen anzusprechen, das nicht existiert oder dessen Name falsch angegeben wurde.

FILE NOT OPEN

Es wurde versucht, ein vorher nicht mit OPEN geöffnetes File anzusprechen, z.B.:

```
10 REM FALSCHE DRUCKERANSPRACHE
20 PRINT#1,"DATEI"
```

Es muß folgende OPEN-Anweisung eingefügt werden:

```
15 OPEN 1,4
```

FILE OPEN

Diese Fehlermeldung tritt auf, wenn Sie versuchen, ein bereits geöffnetes File mit einer OPEN-Anweisung anzusprechen, z.B.:

```
OPEN 1,4: PRINT#1, ...
OPEN 1,4: PRINT#1, ...
```

Entweder müssen Sie hier die erste OPEN-Anweisung durch CLOSE 1 auch wieder schließen, bevor Sie ein zweites Mal OPEN 1,4 verwenden, oder die zweite OPEN-Anweisung muß entfallen.

FORMULA TOO COMPLEX

Ein verwendeter String-Ausdruck ist zu umfangreich, oder eine Formel enthält zuviele Klammern.

ILLEGAL DIRECT

Eine nicht direkt ausführbare Anweisung (z.B. INPUT) wurde im Direktmodus verwendet.

ILLEGAL QUANTITY

Das Argument einer Funktion liegt außerhalb des zulässigen Bereichs.

LOAD

Ein Fehler ist beim Laden eines Programms von der Kassette aufgetreten.

NEXT WITHOUT FOR

Diese Fehlermeldung tritt auf, wenn Schleifen sich überkreuzen bzw. die Laufvariable einer NEXT-Anweisung nicht der bei FOR verwendeten entspricht, z.B.:

```
10 FOR I = 1 TO 10
20 PRINT I,I*2
30 NEXT J
```

oder:

```
10 REM NEXT WITHOUT FOR
20 FOR I = 1 TO 10
30 FOR J = 1 TO 5
40 PRINT I,J,I*J
50 NEXT I
60 NEXT J
```

NOT INPUT FILE

Sie haben versucht, durch INPUT# oder GET# Daten aus einem File zu lesen, das nur zum Schreiben geöffnet wurde.

NOT OUTPUT FILE

Daten sollten mit PRINT# an ein File übergeben werden, das nur zum Lesen geöffnet wurde.

OUT OF DATA

Mit der READ-Anweisung sollten mehr Daten eingelesen werden, als in den DATA-Zeilen vorhanden sind.

OUT OF MEMORY

Es ist kein RAM-Bereich mehr verfügbar. Diese Fehlermeldung kann folgende Ursachen haben:

1. Das Programm oder die Variablen brauchen mehr Speicherplatz, als vorhanden ist.
2. Es wurden zu viele FOR NEXT-Schleifen oder Unterprogramme ineinander verschachtelt.
3. Es wurden zu viele Klammern geöffnet.

OVERFLOW

Das Ergebnis einer Berechnung ist größer als die größte zulässige Zahl.

REAL-Zahlen müssen im Bereich von
-2.93873588E - 39 bis
1.70141183E + 38 liegen.

REDIM'D ARRAY

Es wurde versucht, ein Feld zum zweiten Mal zu dimensionieren.

REDO FROM START

In einer INPUT-Anweisung wurden numerische Werte erwartet, Sie haben jedoch Strings eingegeben. Dieser Fehler führt nicht zum Programmabbruch. Das Programm wird fortgesetzt, wenn nach der Fehlermeldung Zahlen eingegeben werden.

RETURN WITHOUT GOSUB

Eine RETURN-Anweisung wurde im Programm gefunden, zu der die entsprechende GOSUB-Anweisung fehlt.

Dieser Fehler entsteht z.B. dann, wenn in ein Unterprogramm mit GOTO hineingesprungen wird, oder das Unterprogramm nach dem Hauptprogramm steht, und das Hauptprogramm nicht mit END abgeschlossen wurde.

STRING TOO LONG

Sie haben versucht, einem String mehr als 255 Zeichen zuzuweisen.

SYNTAX

Diese wohl am häufigsten auftretende Fehlermeldung tritt auf, wenn eine Anweisung vom Computer nicht erkannt werden kann, z.B. durch einen Schreibfehler, Fehlen einer Klammer oder eines Kommas usw.

TYPE MISMATCH

Es wurde versucht, einer Stringvariablen eine Zahl zuzuordnen bzw. umgekehrt.

UNDEF'D FUNCTION

Es wurde Bezug genommen auf eine selbst definierte Funktion, die noch nicht mit DEF FN definiert wurde.

UNDEF'D STATEMENT

Diese Fehlermeldung wird dann ausgegeben, wenn zu einer nicht vorhandenen Zeilennummern verzweigt werden soll.

VERIFY

Das Programm im Arbeitsspeicher stimmt nicht mit dem angesprochenen auf der Diskette bzw. Kassette überein.

Literaturverzeichnis

Weitere Bücher zum Commodore 64:

1. COMMODORE BUSINESS MACHINES, Inc. (Hrsg.): "Commodore 64 Programmer's Reference Guide". Howard W. Sams u. Co., Inc., Indianapolis (1983).

Sehr umfangreiche (englische) Dokumentation zu diesem Computer, u.a. mit BASIC- und Maschinensprache-Einführung. Ausführliche technische Beschreibung der verwendeten Microchips.

2. COMMODORE GmbH (Hrsg.): "Commodore Microcomputer Handbuch". Commodore GmbH, Frankfurt (1982).

Die dem Commodore 64 beigelegte, leider sehr spärliche Anleitung zu diesem Computer.

3. ANGERHAUSEN,M./BECKER,U./ENGLISCH,L./GERRITS,K.: "64 Intern. Das große Buch zum Commodore 64". DATA Becker GmbH, Düsseldorf (1983).

Das Buch enthält das komplette ROM-Listing, außerdem vielfältige Informationen über den Rechneraufbau. Programme überwiegend in Maschinensprache, deshalb eher für den erfahrenen Programmierer.

4. ANGERHAUSEN,M./ ENGLISCH,L./ GERRITS,K.: "64 Tips und Tricks: Eine Fundgrube für den Commodore 64-Anwender". DATA Becker GmbH, Düsseldorf (1983).

Zahlreiche Programme, sowohl in BASIC, als auch in Maschinensprache. Zusätzlich Informationen über andere Programmiersprachen und das CP/M-System zu diesem Computer.

5. ELSING,J./HERRMANN,D.: "Wirtschaft auf dem Commodore 64". IWT-Verlag, Vaterstetten (1983).

Programmsammlung mit Erläuterung der wirtschaftlichen Grundlagen, z.B.: Lohnsteuerberechnung, Zinsrechnung, Optimierung.

6. ELSING,J./STERNER,H./WAGNER,A.: "Grafik auf dem Commodore 64. Anregungen und Erläuterungen in BASIC." IWT-Verlag, Vaterstetten (1983).

Leicht verständliche Einführung mit vielen Programmen zu den Grafikmöglichkeiten des Commodore 64, u.a. ein Hilfsprogramm zur Sprite-Erstellung auf dem Bildschirm.

Für alle im IWT-Verlag erschienenen Bücher zum Commodore 64 sind auch Disketten erhältlich.

Weitere BASIC-Literatur:

7. MENZEL,K.: "BASIC in 100 Beispielen". Teubner, Stuttgart (1982).

Außer einer kurzen Darstellung der Grundlagen der BASIC-Programmierung enthält das Buch vor allem eine Sammlung von 100 kurzen Programmen mit Beschreibung.

8. SCHUPP,W.: "Schüler programmieren in BASIC." Schoeningh, Paderborn (1980).

Ein Lehr- und Übungsbuch mit 150 Programmbeispielen und 260 Übungsaufgaben.

Stichwortverzeichnis

A

ABS 56
Absoluter Wert 56
Abschwellen 118
Adresse 86
Algorithmus 18 f.
AND 63 ff., 168
Anführungszeichen 25 f., 31,59
Anhängen von Daten 77 f.
Anschlag 114, 118
arithmetische Operationen 25 ff.
ASC 59
ASCII-Code 59
ATN 56
Ausgabe 25 f.
Aushalten 118
Aussage 63 f.

B

BASIC 16, 25 ff., 41 ff.
BASIC-Speicher verkleinern 159 f.
Bedingung 35
Bildschirmspeicher 87 ff.
Bildschirmspeicher verschieben 157 f.
Bit 165 ff.
Byte 165 ff.
Binärzahlen 166 f.

C

CHR$ 59
CLOSE 73
CLR-Taste 144
CONT 336
COS 56
Cosinus 56
Cursorsteuerung 143 ff.

D

DATA 41
Datei 71 f.
Datenverarbeitung 15
Datenverwaltung 77 f.
DEF FN 57, 188
Dialog 30
DIM 53
Direkt-Modus 26
Disk(ette) 37, 69 f., 75 f.
Dollar-Zeichen 59 ff.
Doppelpunkt 32
Dreiecks-Welle 116
Dual-System 166 f.
Dezimal-System 166
Dezimalzahlen 45, 166

E

Eingabe 29 ff.
END 42
EXP 56
Exponentielle Notation 46 f.

F

falsch 63 f.
Farbcodes 314
Farbspeicher 87 ff.
Fehlermeldungen 336
Feld 50 ff.
File 71
Filter 127
Flußdiagramm 17 ff.
FN 57
FOR ... NEXT 44
Formatierfunktionen 56
FRE 62
Freier Speicher 62
Frequenz 113, 116, 125
Funktionen 56 ff.
Funktionstasten 160 f.

G

Ganze Zahlen 45
Gerätenummer 69 f.
GET 35
GET# 75
gleich 63
GOSUB 49
GOTO 35
Grafik 107 ff.
größer als 63
größer oder gleich 63
Groß/Kleinschrift 151
Großschrift/Grafik 151

H

Hauptprogramm 48
Hexadezimal-System 169 ff.
High-Byte 167 ff.
Hintergrundfarbe 90
Hochauflösende Grafik 95, 107 ff.
HOME-Taste 144
Hüllkurve 118

I

IF ... THEN 35
Index 50 f.
INPUT 30
INPUT# 74
Instrumente 116
INT 49, 57
Integer-Variablen 45 f.
invers 145

K

Kassettenrecorder 37, 69f.
Klang 113
kleiner als 63
Komma 27
Kommentar 42

L

Laden von Daten 74, 76
Laden von Programmen 37, 70
Laufvariable 44
Lautstärke 113
Leerzeichen 27, 56
Leerzeile 28
LEFT$ 60
LEN 59
LET 31
linearer Programmablauf 33
LIST 29
LOAD 37, 70
Löschen 28, 62
LOG 57
Logarithmus 57
logische Filenummer 72
logische Operatoren 63 ff.
logische Variablen 63
Low-Byte 167 ff.

M

Mantisse 47
Maschinensprache 62
Mathematische Funktionen 58
Mehrfarben-Modus 90
Melodie 114 f.
MID$ 60
Musik, einstimmig 113 ff.
Musik, mehrstimmig 127 ff.
Musik-Übersicht 138 f.

N

Name von Programmen 37
Name von Variablen 30 f., 45
NEW 28
NEXT 44
Nibble 165 ff.
NOT 63 f.
Noten 113 ff.
Noteneingabe 125
Numerische Funktionen 56 f.

O

ODER 63
ON ... GOTO 50
ON ... GOSUB 50
OPEN 71 f.
OR 63 ff.

P

PEEK 86
POKE 86
POS 56
Potenz 56, 166
PRINT 25 ff.
PRINT# 72
Programm 15
Programmiersprache 16
Prozent-Zeichen 45 f.
Puffer s. Speichertabelle

Q

Quadratwurzel 58

R

Rahmenfarbe 90
RAM 85
READ 41
Real-Variablen 45
Rechteck-Welle 116
REM 42
RESTORE 42
RETURN 49
RETURN-Taste 25 f.
RIGHT$ 60
RND 57
ROM 85 f.
RUN 26
RUN/STOP-Taste 49
Rundung 57

S

Sägezahn-Welle 116
SAVE 37, 69
Schleifen 44 f.
Schwingung 208
Semikolon 27
sequentiell 77
SGN 58
Signum 58
SIN 58
Sinus 58
Sound-Chip 113 ff.
Sortieren 238 ff.
Space 56
SPC 27, 56
Speichern von Daten 71 ff, 75 f.
Speichern von Programmen 37, 69
Speicherorganisation 85 ff.
Speicherplatz 46, 53, 85f.
Spiele 262 ff.
Sprites 95 ff.
Sprites-Übersicht 105 f.
Sprung 35 f.
SQR 58
ST 81
STEP 44
Steuerzeichen 145
STR$ 61
Strings 32 ff., 45 ff., 59 f.
String-Variablen 32,45,59
Strukturierte Programmierung 21
Syntax 16
SYS 62

T

TAB 27, 56
Tabellen 306
Tabellenverarbeitung 50 f.
TAN 58
Tangens 58
Tastatur 26
Teilstring 60 f.

Text 32
Textverarbeitung 59 f.
THEN 35
TI 32
TI$ 32, 176
TO 44
Tondauer 113 ff.
Tonerzeugung 113
Tonhöhe 113
Tonleiter 125

U

Uhr 176
Umwandlung von Strings in Zahlen 61
Umwandlung von Zahlen in Strings 61
Umwandlung zwischen Zahlensystemen 196
ungleich 63
Unterprogramme 48 f.
USR 62

V

VAL 61
Variable 30f., 45 ff.
Vergleichs-Operatoren 63
VERIFY 70 f.
Verknüpfung, logische 63 f.
Verknüpfung von Strings 61
Verzweigungen 33 ff., 50
Video-Chip 95 f., 150 f.
Vordergrund-Farbe 90

W

wahr 63 f.
Wahrheitswert 63 f.
Wahrscheinlichkeit 194
WAIT 62
Wellenform 114
Wissenschaftliche Notation 46 f.

Z

Zahlenwert 32
Zeichenfarbe 107
Zeichenfolge 32
Zeichenkette 32 f., 59 f.
Zeichensätze 148 ff., 151
Zeichensatz ändern 148
Zeiger 42,160
Zeilennummern 26
Zufallszahlen 57, 91
Zusammenfassung, Musik 138 ff.
Zusammenfassung, Sprites 105 ff.
Zuweisung 29 ff.
Zwischenspeicher 31

Programmkassetten und -disketten
zu diesem Buch

BASIC AUF DEM COMMODORE 64

Warum sich Arbeit machen, die andere schon für Sie erledigt haben ?

Die Programme - zum Teil in erweiterter Form - aus diesem Buch können Sie auf Disketten oder Kassetten zusammen mit einer ergänzenden Beschreibung beziehen.

BASIC-Diskette: Best.-Nr.882-22-101 DM 78.00
BASIC-Kassette: Best.-Nr.882-22-501 DM 78.00

Außerdem liefern wir:

Grafik-Diskette:Best.-Nr.880-22-101 DM 58.00
Grafik-Kassette:Best.-Nr.880-22-501 DM 58.00
Wirtschaft-Disk:Best.-Nr.881-22-101 DM 58.00
Wirtschaft-Kass:Best.-Nr.881-22-501 DM 58.00

Diese Preise enthalten die gesetzliche MwSt.

Ihre Bestellung richten Sie bitte an:

IWT Software Service GmbH

Altenberger Str.23 b
5093 Burscheid
Tel: 02174/62815
Telex: 5213989 iwt d

Verkaufsbüro:
Dahlienstraße 4
8011 Vaterstetten
Tel: 08106 31017
Telex : 5213989 iwt d



Grafik auf dem Commodore 64

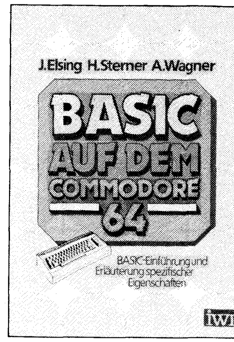
Von Jürgen Elsing, Heinz Sterner und Annette Wagner, alle Bochum, 138 Seiten, 1 Folie, 1983, **ISBN 3-88322-027-2, Spiralh., DM 38,–**

Ausgehend von einfachen Grafiken mit den „fest eingebauten" Grafik-Zeichen des Commodore 64, führt das Buch systematisch zu den anspruchsvolleren grafischen Gestaltungsmöglichkeiten dieses Computers, illustriert jeweils durch ein typisches Beispiel-Programm. Unter anderem findet man ein Hilfsprogramm zur Erstellung der „Sprites"-Grafik, einer besonderen Grafik-Möglichkeit dieses Computers zur Erzeugung beweglicher Figuren.
Zahlreiche Tabellen und Übersichten runden die Darstellungen ab. Die beiliegende Folie hilft bei der Erstellung von Sprite-Figuren.

Wirtschaft auf dem Commodore 64

Von Jürgen Elsing, Bochum und Dietmar Herrmann, Anzing, ca. 200 Seiten, mehrere Abbildungen, 1983
ISBN 3-88322-030-2, Spiralh., ca. DM 38,–

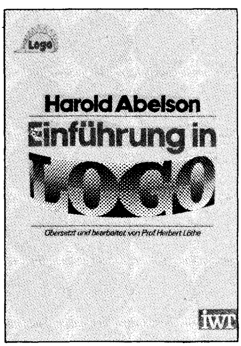
Dieses Buch enthält 40 Programme aus den Bereichen Finanzmathematik, Unternehmensforschung (Operationsresearch) und Betriebswirtschaft. Einige Einzelthemen: Zins- und Rendite-Berechnungen, Renten- und Tilgungsberechnungen, Optimierungs- und Entscheidungstheorie, Investitionsrechnung, Lohnsteuer und Abschreibungen.
Alle Programme werden durch zahlreiche, durchgerechnete Beispiele erläutert.



BASIC auf dem Commodore 64
BASIC-Einführung und Erläuterung spezifischer Eigenschaften

Von Jürgen Elsing, Heinz Sterner und Annette Wagner, alle Bochum, ca. 352 Seiten, 1983, **ISBN 3-88322-029-9, Spiralh., ca. DM 56,—**

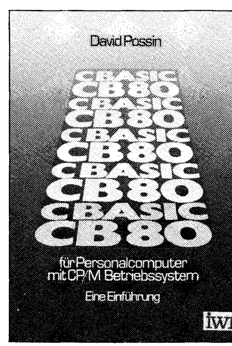
Aus dem Inhalt: Allgemeine Grundlagen — Programmieren in BASIC — Speicherorganisation — Sprites und hochauflösende Grafik — Musik-Programmierung — BASIC-Programmsammlung — BASIC-Übersicht — Tabellen — Stichwortverzeichnis — Literaturverzeichnis — Sachgebiete — Kalender — Mathematik — Statistik — Finanzmathematik — Grafik — Musik — Spiele — Simulation — Lernen.



Einführung in Logo

Von Harold Abelson und Prof. Herbert Löthe, Rommelshausen
196 Seiten, 1983, **ISBN 3-88322-023-X, Spiralh., DM 42,–**

Logo ist eine leicht zu erlernende Computersprache, die in USA bereits bei Kindern erfolgreich eingesetzt wird. Sie ist aber keine "Spielsprache" sondern besitzt wichtige Eigenschaften moderner Programmiersprachen.
Ein wesentlicher Teil von Logo ist die "Igel-Grafik". Mit einfachen Befehlen und Programmen können auf dem Bildschirm komplexe Zeichnungen erstellt werden. Logo ist eine interpretierende Programmiersprache, so können alle Funktionen und Programme ohne Wartezeit sofort ausgeführt werden.



CBASIC — CB 80
für Personal-Computer mit CP/M-Betriebssystem — Eine Einführung —

Von David Possin, Hildesheim, ca. 200 Seiten, 1983
ISBN 3-88322-034-5, Spiralheftung, ca. DM 56,–

Aus dem Inhalt: Basisinformation zu CBASIC — Programmaufbau — Datentypen und Deklarationen — Anwenderdefinierte Funktionen — Ausdrücke und Zuordnungen — Vordefinierte Funktionen — Der Ablauf der Steueranweisungen — Ein-/Ausgabe-Verarbeitungsbefehle — Dateiverarbeitungsanweisungen — Formatierte Ausgaben — Der Compiler Betrieb — LK 80 (Linker) — Die Programmbibliothek von CB-80 — CBASIC II und CB 80 — Syntaxdiagramme — Diverse Anhänge.

CBM 8050 - DOS - Listing
Betriebssystem im Detail

Von Dr. Ruprecht, München, 168 Seiten, zahlreiche Programme,

ISBN 3-88322-015-9, Ringordner, DM 104.--

In einer Art überdimensionalen Kreuzworträtsel ist es dem Autor gelungen, das Betriebssystem der CBM 8050 zu "knacken". Dieses Werk ist nicht nur für "tätige" Programmierer, sondern auch für "nur neugierige" Computer-Fans: Sie bekommen hier Einblick in das Zusammenwirken von zwei 6502-Prozessoren. Hinzu kommt der Datenverkehr über PIAs mit dem Rechner. Dieses System arbeitet dann sehr selbständig zeitlich parallel. Ein Leckerbissen für alle, die ihren Commodore noch besser nutzen wollen.





Herrmanns CBM Programmsammlung in Basic
Band 1: Spiele, Knobeleien und Simulationen

Von Dietmar Herrmann, Anzing, 232 Seiten, 2. Auflage 1983
ISBN 3-88322-013-2, kart., DM 32,–

Der erste Band einer neuen Reihe, die mit zahlreichen Programmen für Spiele und "ernsthafte' Themen den Commodore-Computer dem Benutzer näherbringt. Dietmar Herrmann hat aus seiner Schulpraxis heraus Programme entwickelt, die das Lernen und Spielen mit dem Computer zum Vergnügen machen.



Herrmanns CBM Programmsammlung in Basic
Band 2: Wirtschaft

Von Dietmar Herrmann, Anzing, 204 Seiten, 1983
ISBN 3-88322-014-0, kart., DM 32,–

Jetzt wird es ernst: Hier wird Ihnen gezeigt, wie Sie Ihren Commodore-Computer für sich arbeiten lassen können: Er hilft Ihnen z.B., Ihren Lohnsteuer-Jahresausgleich oder die Einkommensteuer-Erklärung zu erledigen, zeigt Ihnen grafisch, wohin die Staatsverschuldung geht — oder auch Ihre eigene, berechnet Ihre Zinsen (Soll oder Haben) auf der Bank, oder wie Sie Ihr Haus finanzieren können.



Herrmanns CBM Programmsammlung in Basic
Band 3: Mathematik

Von Dietmar Herrmann, Anzing, ca. 200 Seiten, 1983
ISBN 3-88322-016-7, kart., ca. DM 38,–

Aus dem Inhalt: Arithmetik — Zahlentheorie — Kombinatorik — Algebra — Geometrie — Numerische Mathematik.

Wirtschaft auf dem APPLE II / IIe

Von Jürgen Elsing, Bochum und Dietmar Herrmann, Anzing, ca. 200 Seiten, mehrere Abbildungen, 1983
ISBN 3-88322-035-3, Spiralh., ca. DM 38,–

Dieses Buch enthält 30 Programme aus den Bereichen Finanzmathematik, Unternehmensforschung (Operationsresearch) und Betriebswirtschaft. Einige Einzelthemen: Zins- und Rendite-Berechnungen, Renten- und Tilgungsberechnungen, Optimierungs- und Entscheidungstheorie, Investitionsrechnung, Lohnsteuer und Abschreibungen.
Alle Programme werden durch zahlreiche, durchgerechnete Beispiele erläutert.



Basic aus der Praxis
Typische Programmbeispiele für alle Berufe

Von Karl-Wilhelm Hillerkus, Lübbecke, ca. 220 Seiten, 1983
ISBN 3-88322-031-0, Spiralheftung, ca. DM 40,–

Dieses Buch enthält 30 lauffähige Programme aus den Arbeitsbereichen: Suchen – Schreiben – Rechnen – Sortieren. Sie sind an keinen Rechner gebunden, da sie unter CP/M und MBasic geschrieben sind. Sie entstanden aus der praktischen Arbeit heraus und haben sich deshalb bereits bewährt. Auch der "Newcomer" kann sie ohne Schwierigkeiten einsetzen. – Ein weiterer Band soll folgen.

In Vorbereitung: **Basic aus der Praxis**
Programmbeispiele für kaufmännisch orientierte Berufe
Von Karl-Wilhelm Hillerkus, Lübbecke, ca. 220 Seiten, 1983
ISBN 3-88322-042-6, Spiralh., ca. DM 40,–

SuperCalc im Einsatz

Von Ludwig Dodt, Frankfurt, ca. 200 Seiten, 1983
ISBN 3-88322-040-X, Spiralheftung, ca. DM 48,–

Aus dem Inhalt: Einleitung – Laden des SuperCalc-Programms – Der SuperCalc-Bildschirm – Das erste Arbeitsblatt – Speichern, Abrufen, Drucken – Ändern und Erweitern des Arbeitsblatts – Die Rechenfunktionen – Besondere Feldformate – Aufteilung des Arbeitsblatts – Verknüpfung von SuperCalc-Arbeitsblättern – Übertragen von Daten in andere Programme – Weitere Arbeitshilfen – Besonderheiten bei verschiedenen Mikrocomputern – Anwendungsbeispiele – Stichwortverzeichnis.





Computertechnik für Manager
Organisations-Manual

Von Emil A. Widmer, Zug, Schweiz. Mit einem Vorwort von Dr. rer. pol. Peter G. Rogge, 222 Seiten mit zahlreichen Abb. und Organisationsschemata, 1981. Vertriebsrechte für alle Länder mit Ausnahme der Schweiz.
ISBN 3-88322-019-1, geb. DM 128,–
ISBN 3-88322-020-5, Ringordner DM 158,–
Best.-Nr. IWT 205-X Zusätzliche Arbeitsblätter und Organisationsformulare, Satz DM 48,–

Dieses Fachbuch wurde innerhalb kürzester Zeit in der Schweiz zu einem Bestseller. Es gibt dem Manager das Rüstzeug schnell und umfassend den EDV-orientierten Ablauf des Betriebes in den Griff zu bekommen. Das Buch wurde von der Beratungsfirma Interplamar aus der Praxis für die Praxis entwickelt. Durch die Arbeitsblätter kann das Organisationsmanual in allen Bereichen eines Betriebes eingesetzt werden.

Fordern Sie unseren ausführlichen Sonderprospekt an!

In Vorbereitung:
Computertechnik für Manager II: Management-Informationssysteme
ISBN 3-88322-021-3, geb., ca DM 128,–
ISBN 3-88322-022-1, Ringordner, ca. DM 158,–

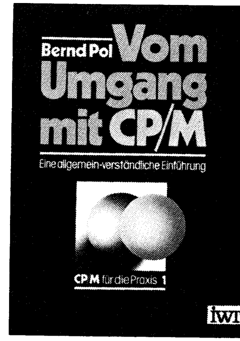
Zu diesen Titeln gibt es Disketten und/oder Kassetten. – Bitte Sonderkatalog anfordern!



Wörterbuch der Computerei, E-D / D-E
mit Erläuterung der wichtigsten Begriffe und Fehlermeldungen

Von Dipl.-Ing. Günther Daubach, Burscheid, 2. erw. Auflage 1983
ISBN 3-88322-026-4, kart. DM 32,–

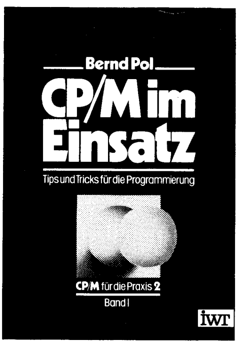
Wer hat nicht bereits verzweifelt vor seinem Personalcomputer-Manual gesessen und versucht, das "Computerchinesenglisch" zu verstehen? Hier hilft jetzt das Wörterbuch der Computerei mit seinen über tausend Begriffen in beiden Sprachen. Außerdem sind die wichtigsten Begriffe zusätzlich erklärt. Ein handliches Nachschlagewerk für jeden, der sich mit Computerei beschäftigt.



CP/M für die Praxis, Band 1:
Vom Umgang mit CP/M - Eine allgemeinverständliche Einführung

Von Bernd Pol, Stuttgart, ca. 350 Seiten mit zahlreichen praktischen Beispielen, 1982, **ISBN 3-88322-004-3, geb. DM 48,–**

Am Anfang dieser Reihe über das neue Betriebssystem CP/M steht eine allgemeinverständliche Einführung. Dem Charakter eines solchen Buches gemäß ist es experimentierend angelegt und führt den Leser im ständigen Kontakt mit dem Computer Schritt für Schritt zu einer umfassenden Übersicht bis hin zur Beherrschung des Systems auch bei Fehlfunktionen.

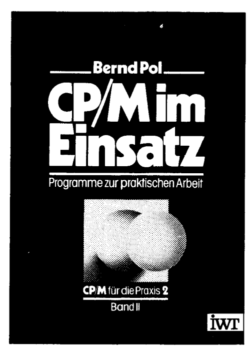


CP/M für die Praxis, Band 2/I:
CP/M im Einsatz -- Tips und Tricks für die Programmierung

Von Bernd Pol, Stuttgart, ca. 250 Seiten. mit zahlreichen praktischen Beispielen, 1983, **ISBN 3-88322-006-X, geb., ca. DM 48,–**

Dieser Band beschreibt alle wichtigen Einzelheiten des BDOS-Kerns und der CBIOS-Systemschnittstelle, die man für den praktischen Einsatz wissen muß und Hinweise zur Fehlerverminderung und -verhütung. Weitere Punkte: Aufbau eines CBIOS-Systems – Standardroutinen zur zeichenorientierten Ein- und Ausgabe – Standardroutinen zum Umgang mit Dateien – Fehlerbehandlung unter CP/M – Erweiterungsmöglichkeiten des CP/M-Betriebssystems – Kompatibilitätsfragen zu MP/M u.a.

CP/M für die Praxis, Band 2/II
CP/M im Einsatz – Programme zur praktischen Arbeit



Von Bernd Pol, Stuttgart, ca. 360 Seiten mit zahlreichen ausgearbeiteten Programmen, 1983, **ISBN 3-88322-032-9, geb., ca. DM 68,–**

Dieses Buch enthält eine Fülle ausführlich kommentierter Programme, die die in 2/I dargestellten Konzepte verwirklichen und ein vollständiges Paket zur Arbeit auf Assembler-Ebene unter CP/M darstellen.
Im einzelnen wird geboten:
- ein Paket mit allgemein benötigten Hilfsprogrammen
- eine Erweiterung der Systemschnittstelle zum Abfangen aller BDOS-Fehler unter CP/M 2.2 ähnlich wie bei CP/Mplus und MP/M II
- ein Paket mit Programmen zum zentralisierten Dateizugriff
- verschiedene Dienstprogramme zur Erweiterung des CP/M-Systems und zum unmittelbaren Diskettenzugriff

Die Programme werden ausführlich kommentiert in einer strukturierten Zwischensprache wiedergegeben und setzen den relozierenden Makro-Assembler RMAC von Digital Research voraus.

Ein umfangreiches Makropaket zur Übersetzung dieser Programme ist ebenfalls enthalten.

In Vorbereitung: **Band 3: Vom Umgang mit CP/M 86** - Eine allgemeinverständliche Einführung. **ISBN 3-88322-005-1**

Bitte frei- machen

IWT Software Service GMBH
Altenberger Str. 23b

5093 Burscheid

Telefon 0 21 74/6 28 15 - Telex 5 213 989 IWT

Hiermit bestelle(n) ich / wir zur sofortigen Lieferung :

Expl.	Disk	Kassette	Titel	à DM/SFr. inkl. MwSt.
			Grafik auf dem Commodore 64	59,–
			Wirtschaft auf dem Commodore 64	59,–
			Basic auf dem Commodore 64	78,—
	Best.-Nr. 850-22-401	C 64	IWT-Sprite-Komfortkit	98,–
	850-18-401	CBM 30XX	IWT-Komfortkit	98,–
	850-19-401	CBM 40XX	IWT-Komfortkit	98,–
	850-20-401	CBM 80XX	IWT-Komfortkit	98,–

Auftrags-Nr. Datum

Unterschrift

Bitte frei- machen

IWT VERLAG GMBH
Dahlienstraße 4
Postfach 10 02 43
D-8011 Vaterstetten
Post Baldham

Telefon (0 81 06) 4986 - Telex 5213989 IWT

Hiermit bestelle(n) ich / wir zur sofortigen Lieferung die folgenden Titel:

Expl.	Best.-Nr.	Titel	à DM/SFr. inkl. MwSt.
	22027	Grafik auf dem Commodore 64	38,–
	22030	Wirtschaft auf dem Commodore 64	38,–
	22029	Basic auf dem Commodore 64	56,—

Auftrags-Nr. Datum

Unterschrift

J. Elsing, H. Sterner, A. Wagner

BASIC auf dem Commodore 64

Dieses Buch bietet eine systematische Einführung in die Programmiersprache BASIC für den Commodore 64. Außer vielen kleineren Programmen zur Illustrierung der einzelnen BASIC-Anweisungen findet der Benutzer eine umfangreiche Programmsammlung zu den verschiedensten Themenbereichen.

Die besonderen Fähigkeiten des Commodore 64 werden mit vielen Beispielprogrammen erläutert.

Für denjenigen, der die zum Teil recht umfangreichen Programme nicht alle selbst eingeben möchte, stehen die Programme auch auf Kassette oder Diskette zur Verfügung.

ISBN 3-83322-029-9